O SCRUM EVOLUIU

3ª Edição

Agradecimentos

Agradeço aos meus amigos por ouvirem meus “discursos inflamados” sobre agilidade e por tanto me incentivarem a escrever esse livro.

Agradeço às empresas Brasfond, Aon, Fidelity, SQLive , Ultralub Química, Acmos, Tivit, Protege, e Credsystem por terem me proporcionado a oportunidade de atuar com metodologias e frameworks ágeis, em diversos momentos da minha carreira, e ao tio Mark Zuckerberg e sua IA pelas imagens(eu apago o logo da Meta porque não quero passar a falsa ideia de que ela endossa meus artigos) e pela correção ortográfica de vários textos.

Dedico esse livreto ao meu filho, Gui, que com seu olhar de pessoa portadora de autismo às vezes é capaz de ver o que as pessoas neurotípicas não veem.

MARCELLO DIAS
PSM1,PSPO1,PSK1 e SPS

Introdução

Este livro nada mais é do que uma coletânea de artigos que escrevi no LinkedIn.

Alguns assuntos são abordados propositalmente de maneira superficial, como gráficos e ferramentas, para que o livro não fique muito grande e massante. O leitor pode usar o Google para se aprofundar, se assim o quiser.

Eu já fui um crítico ferrenho do Scrum, pois o achava engessado se comparado ao Lean Kanban(Não fui eu que mudei, foi o guia do Scrum).

O Scrum evoluiu , mas muitos Scrum Masters não, ainda propagam pensamentos antiquados, mitos e erros de interpretação.

Ron Jeffries inventor das story points não as utiliza mais.

Se a Review servisse para entregar algo ela se chamaria delivery meeting, o Scrum trabalha com entrega contínua, a Review serve para revisar o que foi feito em um período de tempo encapsulado pela Sprint.

Stakeholders não são contagiosos, não precisam ser afastados da equipe Scrum.

O QA é só um desenvolvedor especializado em testes, não é uma babá.

O Scrum “não deu certo” para você? Atualize-se.

A culpa do tombo não é da bicicleta, é do ciclista, o Scrum é um framework testado e aprovado nas mais diversas situações.

ÍNDICE

A Daily Scrum revolucionou o desenvolvimento de software. 49
A Evolução dos frameworks ágeis e a necessidade de adaptação na era dos microsserviços 106
A fantástica fábrica de certificados. 100
A importância das reuniões do refinamento de backlog. 47
Aqui estão alguns livros que eu recomendo sobre Scrum. 109
As diversas configurações possíveis do time Scrum. 59
Com P.O fraco não tem time bom. 36
Como o Scrum Master Deve Agir em uma Retrospectiva com Clima Tenso? 26
Como selecionar e treinar Product owners? 38
Comparativo de Frameworks de escalonamento do Scrum . 95
Dinâmicas visando fortalecer os valores do Scrum. 78
Doações para entidades que cuidam da criança autista. 115
Estratégias para que o QA não vire o gargalo da Sprint. 66
Hard e soft skills necessários para ser um bom Scrum Master. 11
Histórias de usuário,critérios de aceitação e tasks, tomando como exemplo a Java Pet Store. 60
Integrando Design Thinking ao Scrum. 63
Lidando com membros tóxicos e panelinhas no time Scrum. 54
Não, não era tudo mato!
No Scrum se trabalha com entrega contínua. 19
O ágil de mentirinha,ou Scrum com rédeas. 30
O que compõe um bom Backlog do Produto? 6
O RH como viabilizador da cultura ágil. 21
O Scrum Master deve ser o agente promotor da transformação, ou pode virar uma chacota. 33
O Scrum Master “palpiteiro”. 15
O Time Scrum e a IA: Oportunidades e desafios. 101
Os stakeholders devem participar do refinamento do backlog? 44

Pode-se mudar as histórias do sprint backlog? 45
Quando o QA vira Product Owner. 88
Que tipo de Product owner você é? E como pode evoluir? 41
Scrum para desenvolvimento e Kanban para sustentação. 92
Scrum:É preciso mudar a cultura da empresa. 24
Sobre o Autor 116
Substituição no time Scrum: Saem Story points e Capacity, entram Throughput e Cycle time. 69
Técnicas para priorização do Backlog. 51
Teremos que cancelar a sprint , e agora? 82
Wip, Cycle time, Lead time, Througput , Lei de Little..... 73

O que compõe um bom Backlog do Produto?

Este, talvez, seja o capítulo mais importante desse livro.

O Product Backlog, ou lista de funcionalidades e requisitos do produto, é o elemento principal do Scrum. Ele funciona como uma lista que está sempre mudando e sendo organizada de acordo com as prioridades, contendo tudo que pode ser necessário para o desenvolvimento do produto.

Um bom backlog não é apenas uma lista de tarefas, mas uma ferramenta estratégica que orienta o progresso contínuo do produto, garantindo que a equipe entregue valor de forma incremental.

Vamos conhecer os principais componentes de um backlog bem elaborado.

Vamos falar sobre sua estrutura, os critérios para garantir sua qualidade e as boas práticas para aprimorá-lo constantemente.

1. Componentes de um Bom Backlog do Produto

Segundo a **Scrum.org**, um backlog eficaz deve conter os seguintes elementos:

a) Itens Claros e Bem Definidos

Cada item do backlog (geralmente representado como **User Stories, Épicos ou Tarefas**) deve ser:

-Valioso: Entregar benefício tangível ao usuário ou negócio.

-Independente: Minimizar dependências com outros itens.

-Negociável: Aberto a discussão e adaptação.

-Estimável: Permitir que a equipe avalie seu tamanho e complexidade.

-Pequeno: Dividido em partes gerenciáveis para facilitar a entrega em sprints.

-Testável: Ter critérios de aceitação claros para validação.

b) Priorização Baseada em Valor

O **Product Owner (PO)** é responsável por priorizar os itens com base em:

-Valor para o cliente/negócio (ROI - Retorno sobre Investimento)

-Riscos e dependências.

-Necessidades , tendências do mercado e feedback dos stakeholders.

-Alinhamento com a visão do produto.

Teremos um capítulo somente sobre priorização do Backlog, onde apresentaremos várias técnicas padrões de mercado.

c) Critérios de Aceitação Definidos

Cada item deve ter **critérios de aceitação claros**, que definam quando uma funcionalidade está "pronta" (Definition of Done - DoD). Exemplo:

Abaixo exemplo de critério de aceitação para funcionalidade "Adicionar ao Carrinho".

-O sistema mostra uma confirmação visual (ex.: popup ou ícone atualizado).

-O carrinho exibe o preço total atualizado.

-Itens removidos são excluídos imediatamente.

-Persiste os itens se o usuário sair e voltar (com sessão ativa).

Abaixo um excelente artigo sobre critérios de aceitação:

https://www.atlassian.com/work-management/project-management/acceptance-criteria

d) Refinamento Contínuo

O backlog não é estático, ele evolui com:

-Novas descobertas (feedback de usuários, mudanças no mercado).

-Decomposição de itens grandes.

-Remoção de itens obsoletos.

São recomendadas sessões regulares de refinamento com a equipe para manter o backlog relevante.

2. Boas Práticas para Manter um Backlog Eficiente

a) Transparência e Acessibilidade

-Todos devem entender o backlog.

-Ferramentas como Jira, Trello ou Azure DevOps ajudam na visualização.

b) Balanceamento entre Detalhamento e Flexibilidade

-Itens no topo (próximos sprints) devem estar mais detalhados.

-Itens no final do backlog devem ser mais abstratos, visando manter a flexibilidade.

c) Colaboração entre PO e Dev Team

-O PO define o "o quê" (valor), a equipe sugere o "como".

Discussões frequentes evitam mal-entendidos.

d) Revisão e Adaptação Constante

-O backlog deve refletir aprendizados de cada Sprint Review.

-Mudanças no mercado ou tecnologia podem exigir ajustes.

3. Erros comuns e como evitá-los

Problema	Solução
Backlog muito grande e desorganizado	Refinar e decompor itens regularmente
Priorização inconsistente	Definir critérios claros de valor (ex: ROI, risco)
Falta de clareza nos itens	Garantir user stories bem escritas e critérios de aceitação
Pouca colaboração com a equipe	Envolver devs no refinamento do backlog

4. Metáforas acerca do backlog do produto

Durante meus anos de envolvimento com o Scrum me defrontei com várias analogias a respeito do backlog, a maioria não saiu da minha cabeça, algumas encontrei em livros, outras em fóruns.

a) O Backlog como uma Bússola

Alguns especialistas em Agile comparam o backlog a uma bússola:

-Direção clara: Assim como uma bússola aponta para o norte, o backlog indica o que deve ser feito para alcançar a visão do produto.

-Ajuste contínuo: Se o vento muda (novas demandas), o PO reajusta a rota, mas o objetivo final permanece.

Um backlog bem mantido e priorizado é como uma bússola para a equipe Scrum – ele mantém todos navegando na mesma direção, independente de condições adversas.

b) O Backlog como um Farol

-Ilumina o caminho: Mostra os próximos passos, evitando que a equipe se perca em funcionalidades desnecessárias.

-Avisa sobre perigos: Itens de alto risco ou dependências críticas são como rochas no mar.

c) O Backlog como um Jardim

-Precisa de cuidado constante: Se não for refinado, vira um "matagal" de itens esquecidos.

d) O Backlog como uma caixa de ferramentas

- Os materiais (tarefas) são armazenados e aguardam sua vez de serem utilizados (desenvolvidos) para construir o produto final.

Conclusão

O Backlog dinâmico, transparente e orientado a valor. Deve ser refinado, priorizado e alinhado com a visão do produto, garantindo incrementos úteis a cada sprint.
A chave para um backlog eficaz está na colaboração entre Product Owner, desenvolvedores e stakeholders, focado no valor entregue ao usuário final.

Hard e soft skills necessários para ser um bom Scrum Master.

A função do Scrum Master é crucial para o êxito de grupos ágeis. Mais do que um facilitador, ele atua como um líder servil, removendo impedimentos, promovendo a melhoria contínua e garantindo que o Scrum seja bem compreendido e aplicado.

Tanto a Scrum.org quanto os livros "Scrum Mastery" (escrito por Geoff Watts) e "Coaching Agile Teams" (escrito por Lyssa Adkins) enfatizam a necessidade de um equilíbrio entre habilidades técnicas e habilidades interpessoais para um Scrum Master eficiente.

1. Hard Skills necessários para um Scrum Master

As hard skills estão relacionadas às competências técnicas e ao conhecimento especializado que um Scrum Master deve adquirir e dominar:

a) Domínio do Framework Scrum

-Compreensão profunda dos **eventos Scrum** (Sprint, Daily, Review, Retrospective, Planning).

-Conhecimento dos **artefatos Scrum** (Product Backlog, Sprint Backlog, Increment).

-Entendimento dos **papéis e responsabilidades** (Scrum Master, Product Owner, Developers).

b) Métricas e Ferramentas Ágeis

-Capacidade de acompanhar métricas como **Cycle Time, Througput, Lead time**

-Familiaridade com ferramentas como **Jira, Trello, Azure DevOps**.

-Conhecimento de técnicas de **refinamento de backlog** (User Stories, Estimation Techniques).

c) Conhecimento de Outros Métodos Ágeis e Lean

-Noções de **Kanban, XP (Extreme Programming), Lean**.

-Entendimento de **fluxos de trabalho e melhoria contínua (Kaizen)**.

d) Resolução de Problemas Técnicos e Impedimentos

-Habilidade para identificar e remover **impedimentos técnicos e organizacionais**.

-Capacidade de auxiliar a equipe na **resolução de conflitos de priorização**.

2. Soft skills fundamentais para um Scrum Master.

As habilidades interpessoais, conhecidas como soft skills, são fundamentais para o Scrum Master, pois possibilitam a liderança sem a necessidade de uma autoridade formal. Essas competências são essenciais para inspirar a equipe e fomentar um ambiente de colaboração eficaz.

Antes de tudo, um Scrum Master deve saber reconhecer seus erros, ter humildade suficiente para dizer: “Não sei, vou pesquisar”, e saber pedir desculpas. Um exemplo vale mais que mil palestras, e quinhentas horas de coaching.

a) Liderança Servil

-Foco em **servir a equipe**, removendo obstáculos e facilitando seu crescimento.

-Empoderar os membros da equipe para que sejam autogerenciáveis.

b) Comunicação Eficaz

-Saber **ouvir ativamente** e transmitir informações de forma clara.

-Facilitar discussões produtivas em **cerimônias Scrum**.

-Adaptar a comunicação para diferentes stakeholders (equipe, PO, liderança).

-

c) Facilitação e Mediação de Conflitos

-Conduzir reuniões de forma **neutra e produtiva**.

-Mediar conflitos dentro da equipe, promovendo um **ambiente seguro e colaborativo**.

d) Empatia e Inteligência Emocional

-Compreender as **necessidades e frustrações** da equipe.

-Ajudar no desenvolvimento de **relacionamentos saudáveis** no time.

e) Coaching e Mentoria

-Aplicar técnicas de **coaching ágil** para ajudar a equipe a evoluir.

-**Mentorar** o Product Owner e a organização na adoção de práticas ágeis.

f) Adaptabilidade e Resiliência

-Lidar com mudanças e resistências de forma **proativa**.

-Promover uma **mentalidade de crescimento** (Growth Mindset).

3. Como Desenvolver Essas Habilidades?

-Certificações: Realizar cursos como **PSM (Scrum.org) ou CSM (Scrum Alliance)**.

-Prática Contínua: Aplicar Scrum em projetos reais e buscar feedback.

-Leitura e Estudo: Aprofundar-se em livros como **"Scrum Mastery"-** Geoff Watts e **"Coaching Agile Teams"-**Lyssa Adkins.

-Networking: Participar de comunidades ágeis.

Para finalizar sugiro a leitura do artigo:https://www.scrum.org/resources/blog/28-characteristics-great-scrum-master

Conclusão:Ser um Scrum Master eficiente requer mais do que apenas ter um entendimento básico do framework Scrum. É fundamental encontrar um equilíbrio entre as hard skills, que representam as competências técnicas, e as soft skills, que englobam habilidades como liderança, comunicação e empatia. Como ressaltam Geoff Watts e Lyssa Adkins, o autêntico Scrum Master é aquele profissional que motiva a equipe, elimina barreiras e fomenta um ambiente propício ao aprendizado constante. Invista no aprimoramento dessas competências e transforme-se em um verdadeiro agente de mudança dentro da sua organização!

O Scrum Master “palpiteiro”.

Decidi escrever esse artigo, pois participei em um time Scrum, cujo Scrum Master havia sido desenvolvedor há muito pouco tempo, e pelo jeito, gostava mais da função de desenvolvedor, do que dá de Scrum Master.

Às vezes suas opiniões eram até interessantes, mas isto gerava uma certa animosidade, principalmente com um desenvolvedor menos experiente, que costumava se sentir ofendido.

Vale ressaltar que esse Scrum Master era um “estranho no ninho”, não havia trabalhado como desenvolvedor na empresa.

O papel do Scrum Master é fundamental para a eficácia do time Scrum.

Suas responsabilidades são: remover impedimentos, facilitar cerimônias e garantir que os princípios do Scrum sejam bem compreendidos e praticados. Entretanto, quando o Scrum Master tem um passado de desenvolvedor , surge a dúvida: ele deve contribuir com opiniões técnicas ou se abster para não comprometer a autonomia do time?

O Scrum master deve ser um líder servidor, cujo principal foco é:

-Ajudar todos a compreender e adotar o Scrum da forma correta.

-Atuar como um facilitador para a equipe Scrum e para a organização.

-Remover obstáculos que atrapalham o progresso do time.

-Promover a auto-organização e o pensamento empírico.

Segundo o Scrum Guide 2020 : o Scrum Master não é um gerente de projeto tradicional nem um líder técnico, mas sim alguém que ajuda o time a se tornar mais eficaz. Isso implica não se omitir completamente de discussões críticas, mas também não impor sua visão técnica.

Scrum Master com Background Técnico: Benefícios e Riscos

Scrum Masters com experiência anterior como desenvolvedores podem trazer um valor significativo — mas também desafios. Veja os principais pontos:

Benefícios

-Compreensão Técnica Aprofundada: facilita a identificação de impedimentos reais, especialmente os que envolvem arquitetura, dívidas técnicas ou integração contínua.

-Credibilidade com o Time: a experiência prática aumenta a confiança e respeito, tornando mais fácil a facilitação de debates complexos.

-Facilitação mais Eficiente: ao entender a linguagem técnica, consegue conectar pessoas certas, articular melhores perguntas e promover soluções colaborativas.

Riscos

-Microgerenciamento: pode começar a direcionar decisões ou sugerir soluções prontas, sufocando a autonomia do time.

-Viés Técnico: há o risco de favorecer tecnologias ou abordagens com base em experiências pessoais, sem considerar o contexto atual.

-Confusão de Papéis: o time pode confundir o Scrum Master com um Tech Lead informal, levando à centralização de decisões.

Exemplo: Em uma sprint planning, o Scrum Master intervém repetidamente para dizer como uma funcionalidade deveria ser implementada. Resultado: o time para de propor alternativas, e dependendo do caso ,podem ocorrer até discussões.

Quando um Scrum Master Deve (ou não) opinar, for a de suas responsabilidades?

O Scrum Master pode dar sugestões e orientações, mas o mais importante é sempre ficar atento ao processo, e não ao conteúdo técnico. Por isso, é fundamental saber o momento adequado para fazer essas contribuições.

Situações em que opinar é apropriado:

-Quando explicitamente solicitado pelo time.

-Para alertar sobre possíveis riscos ou impedimentos técnicos (ex.: "Essa solução pode dificultar o deploy contínuo?").

-Em discussões sobre práticas de engenharia ágil, como Definition of Done, integração contínua ou qualidade de código.

-Para desencadear reflexões, não para ditar soluções (ex.: "Já consideraram como isso impacta o desempenho do sistema?").

Situações onde é melhor que ele se cale:

-Em decisões técnicas que cabem exclusivamente ao time de desenvolvimento.

-Quando o Scrum Master interrompe ou domina discussões, limitando a troca entre os desenvolvedores.

-Quando não há convite direto para opinar, principalmente se o time já estiver colaborando efetivamente.

Usando a psicologia:

Em vez de sugerir diretamente uma tecnologia, faça perguntas que levem o time à reflexão: *"Qual seria o impacto de usar essa abordagem em relação à segurança?"*

Um Scrum Master técnico não precisa ficar quieto, mas deve usar sua influência de forma sábia. Seu conhecimento deve ser utilizado para orientar de maneira leve, facilitar de forma inteligente e fazer questionamentos estratégicos, sempre de maneira colaborativa, nunca como uma imposição.

> Mike Cohn, em *Succeeding with Agile*, destaca que o Scrum Master deve ser "um guia na jornada de melhoria contínua", não o detentor das decisões.

Esse equilíbrio é o que faz com que um simples palpite se torne uma contribuição realmente importante, preservando a autonomia da equipe e fortalecendo a auto-organização — um dos pilares do Scrum.

É importante destacar que um Scrum Master pode exercer simultaneamente a função de desenvolvedor, desde que essa condição esteja bem definida.

Conclusão

Sim, opiniões são bem-vindas — mas com moderação, com propósito e com escuta ativa. Um Scrum Master atencioso, que sabe quando se manifestar e, principalmente, quando facilitar a expressão dos outros, pode ser um excelente aliado. A experiência técnica é uma vantagem competitiva, desde que seja usada a serviço da equipe, não como um atalho para o controle. O Scrum Master, atuando como mentor e guia, em vez de um tomador de decisões técnicas, pode ser um catalisador para a construção de equipes mais fortes, autônomas e produtivas.

No Scrum se trabalha com entrega contínua.

Anteriormente, fui um crítico fervoroso do Scrum.
No entanto, não fui eu que mudei, foi o Scrum, ou pelo menos a percepção que todos possuíam sobre o Scrum.
Sempre defendi a constante melhoria, a inspeção e a adaptação.

Sempre optei pela prototipagem ao invés do Waterfall, muito antes da existência do manifesto ágil.
Prototipagem é uma espécie de Scrum sem papéis,responsabilidades, pilares,valores e cerimônias, que dependia muito do talento e do bom senso de cada pessoa, mas já era superior ao waterfall.

A maioria das empresas, erroneamente, associa a entrega do incremento ao término da Sprint, como se a Review fosse um altar para o Deus Agilitos.
Se Agilitos não obtiver seu sacrifício, toda a tribo será atingida por sua ira.
O Scrum nunca teve grande popularidade na Ásia, onde as equipes sempre optaram pelo Kanban, com seus princípios Lean.
No entanto, agora que o Scrum incorporou técnicas do Kanban, e reforçou ainda mais os princípios do Lean, foi como ter o melhor dos dois mundos, passei de crítico a admirador.

TRABALHANDO COM ENTREGA CONTÍNUA NO SCRUM.

A maioria dos times Scrum atuais ainda associa a entrega ao altar de Agilitos(Review), fazendo com que muitos se sintam tentados a se comprometer com o mínimo de "story points".
Hoje em dia, em algumas equipes, necessita-se de um exército de Oompa-Loompas para implementar um simples sistema de cadastro. Até mesmo Waterfall seria superior a isso.

Ao combinar o Kanban com o Scrum, estimar não é o mais importante, você decompõe a estória, quando se tem "quase certeza" de que será viável finalizá-la em uma Sprint(atender o SLE), limita-se a quantidade de PBIŚ na linha de produção (WIP), e o foco muda para a entrega contínua.
Pode se enviar para a produção, antes, ou depois da review, o que não estiver pronto retorna ao backlog do produto, geralmente sendo retomado na próxima Sprint.

Mas vale ressaltar que selecionando o conteúdo do Sprint Backlog através da análise de Troughput e Cycle Time, estabelecendo-se um WIP,

a probabilidade de se entregar tudo o que foi selecionado na Planning aumenta muito.

Vale lembrar que a maioria das metodologias e frameworks ágeis datam da década de 90, onde um incremento quase sempre equivalia a um monolito, hoje com o advento dos microsserviços, muitas vezes, a equipe Scrum trabalha em vários incrementos em uma Sprint, não faz sentido esperar a Review para entregar tudo, quando muitas vezes o incremento gera valor, e é aguardado ansiosamente na produção.

Em um cenário como esse, faz-se ainda mais necessário o controle das dependências, lembrando que existem gráficos que mostram a dependência entre os itens do sprint backlog.

Conclusão

Se a equipe terminou uma história antes, e a entrega gera valor, por que esperar pela review? Se o P.O assim decidir, o incremento pode seguir para a produção antes da review.

Ao deixar de utilizar métricas com alto nível de imprecisão, como os story points, e passar a se focar em throughput, tempo de ciclo e estabelecer um limite de trabalho em andamento (WIP), a chance de completar todas as tarefas durante a Sprint se eleva consideravelmente. Muito mais confiança na montagem do Sprint Backlog!!!

O que não foi apresentado na review ,na Sexta-feira,pode muito bem estar pronto na segunda subsequente(próxima sprint) ,e ir para produção, ninguém morreu, com isso a equipe para de se comprometer com pouco ,e reforça-se a transparência um dos pilares do Scrum.

O RH como viabilizador da cultura ágil.

Quando as empresas adotam o Scrum não estão apenas mudando a forma de trabalhar, mas passando por uma verdadeira transformação cultural. Isso significa que os times precisam ser autogerenciáveis, colaborativos e muito eficientes. Nesse cenário, o papel do Recursos Humanos (RH) fica ainda mais importante, já que é responsável por garantir que os profissionais contratados tenham não só as habilidades técnicas necessárias, mas também os valores e as habilidades interpessoais (soft skills) essenciais para se dar bem em um ambiente ágil.

A alta rotatividade: consequência de contratações mal alinhadas

Um dos maiores problemas em empresas que implementam o Scrum é a alta rotatividade de funcionários, frequentemente causada por falhas no processo de recrutamento. Muitas vezes, o RH recebe demandas vagas ou mal definidas por parte dos gestores, resultando em contratações equivocadas.

"Funcionário mal contratado é futuro funcionário demitido" – Se a pessoa não se encaixa na cultura, nos valores ou nas expectativas do time, sua saída é apenas questão de tempo.

Falta de clareza nas vagas – Empresas muitas vezes buscam um "generalista que sabe um pouco de tudo", quando o que realmente precisam é de um especialista com conhecimento profundo em áreas-chave, todos podem aprender tudo, mas qualidade vem com a experiência, a equipe Scrum deve ter os conhecimentos necessários para realizar suas tarefas com exatidão.

O artigo a seguir fala sobre balancear conhecimento generalista com especialista na equipe Scrum: Balancing Generalists and Specialists– Building Successful Agile Teams - InfoQ .
Os assistentes de IA, hoje, minimizam a necessidade de se preocupar com generalistas, qualidade não é algo negociável, contrate os especialistas que precise, o generalismo (que pouco será utilizado) é fácil de se conseguir.

No Scrum, times eficientes são formados por profissionais que dominam suas disciplinas e colaboram de forma sincronizada. Um desenvolvedor que conhece superficialmente várias linguagens, mas não resolve

problemas complexos em nenhuma delas, pode se tornar um gargalo em vez de um ativo.

O Perigo do "Tecnicamente Maravilhoso" com soft skills ruins.

Um erro comum em processos seletivos é supervalorizar habilidades técnicas e ignorar fit cultural e soft skills. No Scrum, a confiança e o respeito são fundamentais – um profissional extremamente competente tecnicamente, mas tóxico em relacionamentos, pode destruir a dinâmica da equipe.

Serial Killers do Ambiente Corporativo – Existem profissionais que são brilhantes individualmente, mas desastrosos em equipe: arrogantes, inflexíveis ou desrespeitosos. No Scrum, isso é inaceitável.

Soft skills não são opcionais – Comunicação, empatia, colaboração e resiliência são tão importantes quanto habilidades técnicas.

Exames psicotécnicos e fit cultural devem ser prioridade – Um candidato pode ser um gênio da programação, mas se não se alinha com a cultura da empresa, seu desempenho será limitado.

O Empoderamento no Scrum Exige um RH Estratégico.

Para que o Scrum funcione, o RH deve ir além do recrutamento tradicional e atuar como um facilitador da cultura ágil:

Definir perfis com clareza – Trabalhar em conjunto com Product Owners e Scrum Masters para identificar as competências técnicas e comportamentais necessárias.

Priorizar fit cultural e soft skills – Implementar dinâmicas, entrevistas comportamentais e avaliações psicométricas para garantir que o profissional se alinhe com os valores do time.

Evitar a armadilha do generalista – Buscar especialistas que tragam profundidade técnica em vez de superficialidade em múltiplas áreas.

Promover um ambiente de confiança – Garantir que os colaboradores tenham autonomia, mas também responsabilidade e respeito mútuo.

Conclusão: A implementação do Scrum só é bem-sucedida quando o RH assume um papel estratégico, indo além da simples contratação e garantindo que os profissionais tenham técnica e temperamento adequados.

Contratações erradas geram demissões futuras – E no Scrum, onde a colaboração é essencial, um erro de recrutamento pode custar caro.

Habilidades técnicas são necessárias, mas não suficientes – Um "gênio" tóxico pode quebrar a confiança da equipe, enquanto um profissional tecnicamente bom e colaborativo eleva todo o time.

Portanto, o RH não é apenas um apoio burocrático – é o alicerce que sustenta times ágeis de alta performance. Quando o recrutamento é feito com critério, clareza e atenção ao fit cultural, o Scrum se fortalece , e a empresa colhe os frutos de um ambiente produtivo, inovador e harmonioso.

Scrum:É preciso mudar a cultura da empresa.

No Scrum, formam-se equipes competentes, auto-organizadas, multidisciplinares, inspecionadas e empoderadas.
A filosofia é que muitos pensam melhor do que um.
Existe hierarquia, mas baseada em respeito e propósito.

Se a cultura da sua organização se fundamenta em estruturas rígidas de supervisão, uma vasta quantidade de relatórios e planos minuciosos, e se a prioridade maior é a burocracia ao invés do produto, então implementar métodos ou frameworks ágeis poderá ser um grande desafio. O Scrum não substitui o setor de Recursos Humanos, é importante contratar profissionais em quem você possa confiar.

Como dizia Steve Jobs: "Não faz sentido contratar pessoas inteligentes e dizer a elas o que fazer; nós contratamos pessoas inteligentes para que elas nos digam o que fazer!".

Em empresas que adotaram o Scrum de verdade, da maneira correta, com funcionários bem selecionados pelo RH, com Scrum Masters certificados por uma instituição séria como a Scrum.org, por exemplo, o que se nota é que os membros da equipe Scrum incorporam o espírito de dono, uma passa a ser o apoiador da melhoria do outro, viram verdadeiros cães de guarda dos valores da empresa, sentem-se roubados quando alguém faz corpo mole, pois cresce dentro deles o senso de pertencimento, já que passam a ter orgulho do produto que desenvolvem.

A comunicação diária e a inspeção contínua entre pares promovem uma incessante busca pela compreensão abrangente do produto sob sua responsabilidade, além de contribuir para o desenvolvimento profissional. Quando o conhecimento e as informações técnicas são compartilhados, todos se aprimoram, elevando seu valor como profissionais, ao mesmo tempo em que a empresa não se torna dependente de uma única pessoa.

Analise a ilustração anterior em contraste com um modelo onde somente o gerente é encarregado de definir as diretrizes de qualidade, estabelecer prazos e determinar soluções técnicas.

Eu não acreditava nisso, a vivência com o Scrum mudou meu pensamento.
Vi desenvolvedores que "empurravam com a barriga" suas carreiras se converterem em devoradores de livros.

Há que se respeitar os traumas que cada empresa carrega, a palavra empoderamento é a que mais causa medo, quando o foco deveria estar em inspeção, adaptação, transparência e qualidade.

Muitos empresários já foram roubados, contrataram profissionais ruins, jogaram dinheiro fora, principalmente no Brasil. Muitas vezes tem-se que implantar o Scrum logo após a empresa ter caído alguma "PicareTIagem"(empresas picaretas existentes na área de TI), ou seja, com grau de desconfiança nas alturas.

Conclusão: O Scrum aumenta o senso de responsabilidade de todos sobre o produto que desenvolvem e estimula a melhoria contínua.

Como o Scrum Master deve agir em uma retrospectiva com clima tenso?

A Retrô é a cerimônia ideal para que a equipe Scrum analise o que deu certo, o que não teve sucesso e como pode evoluir.

Entretanto, se a atmosfera estiver tensa—particularmente quando um desenvolvedor é alvo de críticas por parte do grupo,
O Scrum Master precisa atuar com sabedoria para converter a discórdia em aprendizado e garantir um ambiente seguro e cooperativo.
Em dois meses, talvez, ninguém se lembre do motivo da discussão, mas se o nível da discussão baixar muito, as consequências podem ser duradouras.

O papel do Scrum Master em uma retrospectiva conflituosa

O Scrum Master é o guardião do processo e deve assegurar que a Retrospectiva seja produtiva, respeitosa e focada em melhorias, não em culpas. Se o time está direcionando críticas de forma hostil a um único membro, o Scrum Master deve intervir de forma estratégica.

1. Estabelecer um ambiente seguro

Relembrar os valores do Scrum (respeito, abertura, coragem): Antes de começar, reforçar que o objetivo é melhorar como equipe, não apontar culpados.

Usar técnicas de facilitação: Dinâmicas como "Start, Stop, Continue" ou "Sailboat Retrospective" ajudam a manter o foco em ações, não em pessoas.

Estabelecer regras claras: Estabelecer expectativas para o diálogo, como consideração mútua e concentração em soluções.

2. Redirecionar a discussão para fatos, não julgamentos

Evitando generalizações: Se alguém afirma "Você está sempre atrasando o andamento da Sprint", o Scrum Master deve reformular a pergunta para "Qual o motivo desse atraso? Como podemos fazer, da próxima vez,para que esse atraso não volte a acontecer?"

Concentrar-se no impacto, e não na pessoa: Fazer a pergunta "De que maneira isso influenciou o time?" ao invés de "Por que você tomou essa ação?"

Empregar a abordagem de "análise de causa raiz": Incentivar a equipe a identificar as razões fundamentais do problema.

3. Garantir que todos tenham voz

Dar espaço a pessoa que esta sendo criticada: Permitir que ele explique seu ponto de vista sem interrupções.

Evitar o "ataque em grupo": Se vários membros estão criticando um colega, o Scrum Master pode intervir com perguntas como:

"O que cada um de nós poderia ter feito diferente?"

"Como podemos apoiar esse membro para evitar o problema no futuro?"

Utilizar a técnica de " feedback construtivo": Encorajar os membros do time a dar feedback específico, comportamental e focado em melhoria.

4. Transformar conflito em melhoria

Perguntar "O que aprendemos com isso?": Incentivar o time a extrair lições, não apenas reclamar.

Definir ações concretas: Se o problema foi um erro técnico, propor pares para revisão de código ou treinamentos.

Criar um plano de ação: Definir passos concretos para evitar problemas semelhantes no futuro.

5.Encerrar focando no futuro

Reafirmar o compromisso do time: Lembrar que a força do grupo vem do todo, que todos devem remar para o mesmo lado.

Agendar acompanhamentos: Se necessário, marcar uma conversa individual com o desenvolvedor para oferecer suporte.

Celebrar os sucessos: Reconhecer e celebrar os sucessos do time, mesmo que pequenos.

Conclusão

Uma retrô tensa pode se transformar em uma chance significativa de desenvolvimento se o Scrum Master funcionar como facilitador, preservando o respeito e guiando a equipe em busca de soluções.
A prioridade deve ser aperfeiçoar os processos, em vez de apontar dedos.
Caso o ambiente continue tenso, um diálogo individual com os participantes pode ser necessária para reafirmar os valores do Scrum.

O ágil de mentirinha,ou Scrum com rédeas.

A corrida pela transformação ágil nas empresas brasileiras tem uma verdade assustadora: boa parte é puro teatro. Empresas adotam, quadros Kanban e cerimônias de Scrum não por convicção metodológica, mas por pressão de mercado. O resultado? O Scrum Master vira um "mini-gerente de projetos", cronogramas rígidos disfarçam-se de "sprints flexíveis", e a cultura Waterfall permanece intocada.

O "Scrum Master" (agora na posição de gerente) solicita atividades e faz atualizações em planilhas, enquanto reuniões se transformam em rituais de avaliação pessoal sem alterações efetivas, uma vez que a grande maioria das decisões ainda necessitam de autorização de níveis superiores.

Scrum Master como um "Pequeno gerente e mestre de cerimônias":

Um verdadeiro Scrum Master atua como um facilitador e eliminador de obstáculos. No entanto, na prática, diversas empresas o transformam em um supervisor de prazos, concentrando-se mais em cumprir cronogramas rigorosos do que em disseminar a cultura ágil.

Ao invés de treinar a corporação sobre os valores e pilares do Scrum , ele se torna um porta-Voz da Hierarquia: Repassa demandas de superiores em vez de fazer a informação e o foco no cliente circularem nos dois sentidos:Top-down e bottom-up.

A tabela abaixo destaca o contraste: Função Autêntica x Distorção

Função Autêntica	Distorção
Remove impedimentos	Cobra entregas
Facilita decisões da equipe	Repassa ordens da gerência
Promove auto-organização	Controla tarefas individualmente

O mindset Waterfall mascarado e a obsessão por previsibilidade absoluta e ainda dominante em várias empresas.
Muitos empresários bem sucedidos(que as vezes começaram sozinhos,ou com pouca colaboração,alguns usando prototipação e não waterfall) seguem a linha de pensamento: “Eu cheguei até aqui sem essa @#!$ de ágil, empoderar é o escambaul!”, mas não enxergam que :

1-Não é tão fácil fazer um colaborador incorporar o ânimo de dono.
2-É muito dificíl se manter a qualidade e o olho no cliente quando os negócios crescem.

Steve Blank, criador do termo "Innovation Theater", alerta: "Empresas realizam iniciativas que parecem inovação, mas criam pouco valor real" .

As consequências para os profissionais envolvidos incluem medo, desligamentos e uma sensação de culpa que muitas vezes não é justa. Muitos Scrum Masters acabam aceitando essa situação por medo de perder seus empregos. Para manter suas posições, eles se transformam em "pequenos gerentes", o que não é o papel original deles.

Conclusão: Ágil é Cultura, não Cosmético, empresas que fingem ser ágeis pagam um preço alto: desperdício de recursos, perda de competitividade, e equipes com alta rotatividade. O caminho para mudança exige consciência cultural e coragem gerencial.
Enquanto organizações buscam "parecer modernas", as que verdadeiramente abraçam a agilidade colhem frutos: produtos alinhados ao mercado, equipes motivadas e inovação sustentável.
A escolha é clara: ser ágil , ou fingir ser e sucumbir a dura realidade.

Página deixada propositalmente em branco

O Scrum Master deve ser o agente promotor da transformação, ou pode virar uma chacota.

O dia a dia de um Scrum master: Desafios e aprendizados

Um Scrum master é um profissional fundamental em times de desenvolvimento de software que adotam o framework ágil Scrum. Seu papel é garantir que o time siga os princípios e práticas do Scrum, removendo obstáculos e melhorando continuamente o processo. No entanto, essa tarefa não é simples e envolve uma série de desafios, principalmente quando se trata de implementar Mudanças.

Implementando mudanças respeitando a cultura da empresa

Um Scrum Master eficaz deve ser capaz de implementar mudanças de forma gradual e respeitosa, considerando a cultura de cada empresa. É essencial entender que cada organização tem sua própria história, valores e maneira de trabalhar, e ignorar esses aspectos pode levar a resistência e até mesmo ao fracasso.

Se um Scrum Master tentar implementar mudanças de forma rápida demais, sem considerar a cultura da empresa, pode enfrentar forte resistência dos funcionários e até mesmo ser demitido. Como Ken Schwaber, co-criador do Scrum, disse: "Um Scrum Master morto é um Scrum Master inútil". Essa frase pode parecer dura, mas reflete a realidade de que um Scrum Master precisa saber navegar pelas complexidades políticas e culturais de uma organização para ter sucesso.

Agora, somplesmente render-se a situação vai levar, a longo prazo, justamente a demissão, ao tornar-se parte da paisagem o Scrum Master passará a ser visto como um inútil.

O Scrum é um framework ágil que se adapta a qualquer tamanho e tipo de empresa. Pode ser implementado em pequenas startups ou grandes corporações, e seu framework é flexível o suficiente para ser adaptado às necessidades específicas de cada time. Além disso, em times menores ou em fases iniciais de adoção do Scrum, é comum que um desenvolvedor acumule o papel de Scrum Master.

O desafio de ser um Scrum Master

Muitos Scrum Masters acabam sucumbindo à pressão de implementar "mudanças com rédeas" , o que pode levar a uma situação em que eles se tornam inúteis ou meros facilitadores de cerimônias do Scrum. Em vez de agir como agentes de mudança, eles podem se limitar a realizar reuniões e seguir procedimentos burocráticos.

Isso ocorre porque, na tentativa de manter seus empregos a qualquer custo, alguns Scrum Masters acabam perdendo de vista seu papel principal: melhorar continuamente o processo e garantir a adesão do time aos princípios do Scrum. Eles podem se tornar tão focados em manter a estabilidade e evitar conflitos que acabam por não cumprir seu papel de forma eficaz.
O Papel do Scrum Master

Um Scrum Master deve ser um líder servidor, capaz de inspirar e motivar o time a trabalhar de forma colaborativa e focada em entregar valor ao cliente. Ele deve ser capaz de identificar obstáculos e trabalhar para

removê-los, melhorar a comunicação dentro do time e garantir que todos estejam alinhados com os objetivos do projeto.
Além disso, um Scrum Master deve ser um facilitador do aprendizado contínuo, ajudando o time a refletir sobre seu próprio processo e identificar oportunidades de melhoria. Isso requer uma combinação de habilidades técnicas, de comunicação e liderança.

O Scrum Master não pode se conformar com impedimentos antigos.

O Scrum Master não pode se conformar com impedimentos **"escritos na rocha"**, ele deve saber mostrar custo e benefício com explanações como:

-"O Sr. sabia que se tivéssemos um ambiente de homologação mais parecido com o de produção, os desenvolvedores seriam mais produtivos, pois não teriam que gerar massa de dados, diminuíriamos também a possibilidade de Rollbacks em ambiente de produção".

— "A ferramenta de desenvolvimento tal é compatível com a última versão do SGBD, tal."

Se o Scrum Master se comportar como um "medrosinho" que se conforma com impedimentos, como coisas naturais da vida, ele/ela virará uma chacota na equipe Scrum e sucumbirá com o tempo.

Conclusão

Ser um Scrum Master é um desafio complexo que requer habilidades e competências específicas. Implementar mudanças de forma respeitosa e gradual, considerando a cultura da empresa, é fundamental para o sucesso.

No entanto, muitos Scrum Masters acabam sucumbindo à pressão e se tornando inúteis ou meros facilitadores de cerimônias. É essencial que os Scrum Masters mantenham seu foco em melhorar continuamente o processo e garantir a adesão do time aos princípios do Scrum.

O Scrum é um framework ágil flexível que pode ser adaptado a qualquer tipo de empresa, e um desenvolvedor pode acumular o papel de Scrum Master em times menores.

Com P.O fraco não tem time bom.

Tem um ditado muito chulo que não irei repetir aqui, sobre pessoas que escolhem seus parceiros levando-se em conta apenas atributos físicos, desprezando caráter e inteligência, e depois reclamam do casamento pelo resto da vida.

Um erro muito comum é escolher o P.O esperando que ele vá aprender com os stakeholders e o scrum master.

O pensamento "Qualquer um pode ser P.O" tem levado a produtos que não acrescentam valor e a intermináveis horas de retrabalho.

O P.O tem que ser "o cara", tem que entender muito da área alvo do produto, tem que ser respeitado pelos stakeholders pelo seu conhecimento, e não ser um simples office boy deles.

O P.O é um maximizador de valor, product owners deveriam tirar certificação Scrum, principalmente aqueles que já foram gerentes.

Gerentes muitas vezes viram péssimos product owners, é preciso passar por uma sessão de desintoxicação, é preciso entender que o respeito vem do conhecimento, diálogo, da sensação de sucesso do grupo e não do efeito carteirada.

Conclusão: Um P.O que não domina a área do produto e não consegue identificar oportunidades resulta geralmente em muito retrabalho e frustração.

"Falhe rápido" não significa jogue o tempo e o dinheiro da companhia no lixo. Um adaptável a qualquer tamanho de empresa.

Como selecionar e treinar Product owners?

O Product Owner, conhecido como PO, desempenha um papel essencial no Scrum. Ele atua como o elo de ligação entre os envolvidos no projeto, os stakeholders, e a equipe de desenvolvimento. A tarefa principal do PO é assegurar que o produto final ofereça o máximo valor, mantendo o backlog devidamente estruturado e alinhado com as demandas do negócio.

Entretanto, a seleção e o treinamento de um bom PO não são processos simples. Muitas organizações encontram desafios ao escolher a pessoa ideal ou ao prepará-la da forma correta para essa posição. Neste artigo, abordaremos como efetuar essa escolha e forneceremos orientações para treinar seu Product owner. Também explicaremos por que a certificação Professional Scrum Product Owner, ou PSPO, pode ser um diferencial importante nesse processo.

1. Como Selecionar um Bom Product Owner?

Antes de começar o treinamento, é importante selecionar a pessoa certa. Algumas qualidades e habilidades essenciais que ela deve ter são:

a) Visão de Negócio
Um PO deve entender profundamente o mercado, os clientes e os objetivos estratégicos da empresa. Ele não é apenas um "gerente de backlog", mas um estrategista de produto.

b) Habilidades de Comunicação
O PO precisa articular claramente a visão do produto, negociar prioridades e mediar conflitos entre stakeholders e o time de desenvolvimento.

c) Capacidade de Tomada de Decisão
Priorizar demandas em um ambiente dinâmico exige julgamento ágil e coragem para dizer "não" quando necessário.

d) Conhecimento Técnico Básico
Embora não precise ser um desenvolvedor, o PO deve entender os desafios técnicos para tomar decisões realistas.

e) Mentalidade de Dono (Ownership)
Um bom PO age como um CEO do produto, assumindo responsabilidade pelo sucesso ou fracasso do que está sendo entregue.

2. Como Treinar Product Owners

Uma vez selecionado, o PO precisa de treinamento contínuo para se manter eficaz. Algumas estratégias incluem:

a) Mentoria e Shadowing
Permitir que o novo PO observe um PO experiente em ação ajuda a entender a dinâmica do papel na prática.

b) Capacitação em Métodos Ágeis
Cursos em Scrum, Lean, Kanban e outras metodologias e frameworks ajudam o PO a entender seu papel no ecossistema ágil.

c) Certificação Professional Scrum Product Owner (PSPO)
A PSPO, oferecida pela Scrum.org, é uma das certificações mais respeitadas para POs. Ela valida o conhecimento em:
-Definição e gestão do backlog do produto
-Priorização baseada em valor
-Colaboração com stakeholders
-Métricas de produto e ROI

A certificação garante que o PO compreende as melhores práticas do Scrum e está preparado para tomar decisões estratégicas.

d) Feedback Contínuo e Retrospectivas
Assim como o time Scrum, o PO deve evoluir com base em feedbacks regulares das partes interessadas e do time de desenvolvimento.

3. Por Que a Certificação PSPO é Importante?

Muitas organizações questionam se a certificação é realmente necessária. A resposta é: depende. Se a empresa busca:
-Padronização de conhecimentos entre seus POs
-Validação externa das habilidades do profissional
-Maior credibilidade perante clientes e stakeholders

Então, a PSPO é um investimento valioso. No entanto, a certificação sozinha não faz um bom PO – experiência prática e soft skills são igualmente essenciais.

Conclusão

Selecionar e treinar um Product Owner requer foco em habilidades técnicas, comerciais e emocionais. A qualificação PSPO pode se destacar ao garantir que o PO tenha o entendimento fundamental para ser um maximizador de valor. Invista recursos em aprendizado contínuo, mentoria e qualificações reconhecidas, mas lembre-se: um ótimo PO é alguém que combina conhecimento teórico, experiência prática e paixão pelo produto.

Que tipo de Product owner você é? E como você pode evoluir?

O papel de Product Owner (PO) está entre os mais complexos e flexíveis no ambiente ágil, demandando uma mistura de competências técnicas, visão de negócios e habilidades de comunicação. Mas, afinal, qual é o seu perfil como PO? E o que fazer para progredir na sua trajetória profissional?

Neste texto, iremos analisar os diversos tipos de Product Owners, identificar áreas que podem ser aperfeiçoadas e estabelecer um roteiro para o desenvolvimento na carreira.

1.Conhecendo o Seu Perfil de Product Owner

Cada Product Owner tem um jeito próprio de trabalhar, que é influenciado pelas suas habilidades, experiências e o ambiente da organização. Aqui estão alguns perfis mais comuns:

a) O PO Tático (Focado na Execução)

Atributos: Ele dá prioridade para entregas rápidas, trabalha com histórias de usuário bem definidas e garante que a equipe Scrum tenha tudo o que precisa para realizar o trabalho.

Vantagens: É excelente em manter o ritmo das sprints e em eliminar obstáculos que possam atrasar a equipe.

Obstáculos: Pode acabar deixando de lado a visão estratégica do produto, esquecendo de alinhar o backlog com os objetivos de longo prazo.

b) O PO Visionário (Estratégico)

Atributos: Enxerga o produto de maneira global, estabelece planos claros e mantém uma conexão próxima com os envolvidos.

Vantagens: Tem uma habilidade excepcional para transmitir a visão do produto e assegurar que a equipe se concentre no que realmente é significativo.

Obstáculos: Em algumas ocasiões, pode desviar a atenção dos aspectos de implementação, resultando em lacunas no backlog.

c) O PO Híbrido (Tático + Estratégico)

Atributos: Combina a execução imediata com uma perspectiva de longo prazo, ajustando-se conforme necessário.

Vantagens: Adaptável e capaz de alinhar o esforço da equipe com as metas da empresa.

Obstáculos: Pode enfrentar sobrecarga, tentando realizar múltiplas tarefas simultaneamente.

d) O PO Técnico (Com Background em Desenvolvimento)

Atributos: Possui um entendimento profundo em tecnologia, o que o ajuda a avaliar e priorizar itens com base em sua complexidade e efeitos.

Vantagens: Melhora a comunicação com os desenvolvedores e elimina gargalos técnicos.

Obstáculos: Pode acabar priorizando soluções técnicas em detrimento da experiência do usuário.

e) O PO Comercial (Focado no Negócio)

Atributos: Tem um conhecimento abrangente sobre o mercado, os clientes e as métricas de desempenho do negócio.

Vantagens: Excelente em validar suposições e garantir um retorno sobre o investimento.

Obstáculos: Pode não levar em conta as dificuldades técnicas ou de experiência do usuário de forma adequada.

Melhorando como Profissional

Para evoluir na função de Product Owner, é essencial:

Aperfeiçoar competências de negócios: Conhecer o setor, o mercado e os consumidores.
Aperfeiçoar a comunicação: Cultivar habilidades de comunicação eficientes para interagir com as partes interessadas e equipes.
O P.O antes de tudo é uma peneira de gostos, intenções e interesses.

Priorizar com foco no valor: Aprender a classificar funcionalidades e requisitos com base no valor para o cliente e nas metas de negócios.

Tomar decisões fundamentadas: Aperfeiçoar habilidades analíticas e de tomada de decisão para fundamentar as escolhas feitas.
Procurar feedback: Solicitar opiniões de partes interessadas, consumidores e equipes para aprimorar a atuação como P.O.

Conclusão

O papel do Product Owner é essencial em equipes ágeis, e o aprimoramento profissional requer habilidades e competências específicas. Compreendendo os distintos tipos de Product Owners e cultivando traços de um P.O eficaz, é possível aumentar a eficácia profissional e fornecer produtos valiosos a clientes e partes interessadas. O livro "The Professional Product Owner"- *Don McGreal and Ralph Jocham* oferece insights valiosos sobre como se aprimorar como P.O e alcançar sucesso em equipes ágeis.

Os stakeholders devem participar do refinamento do backlog?

O Scrum é um framework leve e adaptável aos processos e as necessidades de cada empresa e produto(não confunda-se com Escrumlacho).

Em times que trabalhem com sistemas complexos e legados a participação dos stakeholders que trabalham diariamente com o produto, dos devs que podem olhar nas entranhas dos sistemas, e do PO que criou a visão,e o goal do produto, tem tudo para tornar sessões de refinamento de backlog mais produtivas,ajudando o P.O a ser mais preciso na criação e priorização das histórias.
Outra vantagem dessa estratégia é que pode-se identificar futuros bons P.Oś entre os atuais stakeholders.

Pode-se mudar as histórias do sprint backlog?

Uma ideia errada sobre o Scrum, mas muito difundida, é que depois da planning não se pode mudar nada do Sprint backlog, não estou falando dos detalhes e da decomposição dos PBIs em tarefas.
Estou falando de trocar PBIs mesmo.

Muitos chamam os PBIs de histórias.

Vivendo no Brasil, um país onde a economia, as leis, e as normas mudam diariamente, já passei por muitas situações, onde a flexibilidade do Scrum foi posta a prova.

O que não pode é o objetivo da Sprint ser desrespeitado, só isto.
Itens podem deixar de fazer sentido no meio da Sprint, por diversos motivos, (uma nova regulamentação do governo, uma aquisição da companhia, a perda de um cliente).

É óbvio que ninguém deve ficar de braço cruzado por isto, basta a equipe chegar a um consenso (P.O/Devs), sobre qual item do backlog pode substituir aquele que foi cancelado.
Já participei de Sprints canceladas pelo P. O, porque a Sprint inteira parou de fazer sentido.

Quando surge algo urgente, nunca é uma estória grande, ou pior, um épico, épicos novos tão urgentes seriam oportunidades de mercado raríssimas que até justificariam o cancelamento da Sprint atual, mas em 13 anos isto nunca aconteceu comigo.

Já tivemos que lidar com pequenos itens urgentes, por vezes a equipe foi capaz de acomodar o novo item sem ter que cancelar nada, por vezes pegamos um item que não era essencial e tiramos da Sprint, por vezes aceitamos criar débito técnico.

Conclusão:O importante é o bom senso.

A importância das reuniões do refinamento de backlog.

No mundo da agilidade, as reuniões de refinamento de backlog são fundamentais para garantir que as equipes estejam alinhadas e focadas nas prioridades corretas. Muito subestimadas, essas reuniões desempenham um papel crucial na preparação do backlog para as futuras sprints, assegurando que os itens estejam claros, priorizados e prontos(Ready) para serem trabalhados.

O refinamento de backlog é o processo contínuo de revisão, atualização e priorização dos itens no backlog do produto.
Os itens devem estar claros, concisos, além de estarem alinhados com os objetivos do produto e da empresa. Sem um backlog bem refinado, as equipes podem acabar trabalhando em itens de baixa prioridade ou mal definidos, o que pode levar a desperdício de tempo e recursos.

Não se estima mais nada no backlog, principalmente em story points, ainda mais no backlog, isso é desperdício de tempo e dinheiro.

A frequência ideal para as reuniões de refinamento de backlog pode variar dependendo do tamanho da equipe e da complexidade do projeto. No entanto, uma boa regra geral é realizar essas reuniões uma vez por semana. Isso permite que a equipe mantenha o backlog atualizado e alinhado com as prioridades atuais, sem sobrecarregar os membros da equipe com reuniões excessivas.

Algumas equipes deixam para refinar o backlog durante a cerimônia de planning, mas isso é uma péssima ideia. A planning é um momento crítico para definir os objetivos da sprint e planejar o trabalho a ser realizado. Se a equipe tentar refinar o backlog durante essa cerimônia, pode acabar gastando muito tempo discutindo detalhes de itens que não estão prontos para serem trabalhados, o que pode atrasar o início da sprint.

Por outro lado, a review é uma grande oportunidade para refinar o backlog. Durante a review, a equipe apresenta o trabalho realizado na sprint anterior e recebe feedback dos stakeholders. Esse é um momento ideal para discutir prioridades, esclarecer dúvidas e atualizar o backlog com base no feedback recebido.

Conclusão

Em resumo, as reuniões de refinamento de backlog são fundamentais para o sucesso no desenvolvimento ágil. Realizar essas reuniões uma vez por semana pode ajudar a manter o backlog atualizado e alinhado com as prioridades atuais. Além disso, é importante evitar refinar o backlog durante a cerimônia de planning e aproveitar a review como uma oportunidade para refinar o backlog.
Com um backlog bem refinado, as equipes podem trabalhar de forma mais eficiente e eficaz, entregando valor aos clientes e stakeholders.

A Daily Scrum revolucionou o desenvolvimento de software.

Você pode até não gostar dela.
Mas a **Daily Scrum** revolucionou a indústria do software.

Comecei minha carreira em uma software house. Um dos primeiros clientes que atendi também contratava outra empresa. Nós usávamos "prototipagem"(Eu chamava de método DOITNOWBRO), enquanto a outra software house usava o método waterfall.

Ou seja, eu entregava algo novo quase todo dia, o rapaz da outra software house chegava, ia para um canto afastado, se fechava no seu mundo, em uma época em que não existiam firewalls.

Nós éramos dois "lonely cowboys", o que mudava era o foco, a qualidade e a metodologia.

Pois bem, quatro meses depois ele pediu demissão. E pasmem: ele não havia feito praticamente nada, e o que tinha feito não tinha qualidade alguma. Em pouco tempo, o cliente nos deu exclusividade, ele dizia que éramos informais demais, mas que éramos os únicos que davamos resultado.

As metodologias e frameworks ágeis, sendo o mais bem-sucedido o Scrum, vieram para preencher essa lacuna entre a prototipação e o Waterfall, mas também para dar transparência, não deixar ninguém empacar (dando auxílio mesmo), possibilitar a inspeção, e acabar com as grandes surpresas ruins.

Conforme descrito no Guia Scrum, o objetivo do Daily Scrum é inspecionar o progresso em direção ao Objetivo da Sprint e adaptar o Backlog da Sprint conforme necessário, ajustando o trabalho planejado para os próximos dias.

A Daily Scrum é propositalmente curta, serve para discutir o andamento da sprint, nada impede que ocorram outras reuniões durante o dia, se necessário, mas ter poucas reuniões extras é sinal de um time maduro.

Quinze minutos equivalem a 0,0104166666666667 de um dia, eles não fazem falta nenhuma no seu tempo. Se você é um anti-daily, você não quer é "dar satisfação para ninguém", “HELLO!!!”, nos dias de hoje, lonely cowboys acabam invariavelmente demitidos, e tenha certeza, nunca são os promovidos.

Um dos motivos da Daily é dar a oportunidade de dar e de pedir ajuda, no Scrum remamos para o mesmo lado o tempo todo.

Eu conheço programadores que reclamavam de ficar o dia inteiro sentados e de ter que ficar 15 minutos de pé.

Conclusão: A Daily faz a informação circular, aumenta a confiança e a coesão da equipe e contribui para a melhora profissional de todos, ela também diminui a chance de atrasos durante a sprint.

Técnicas para priorização do Backlog.

Priorização do Backlog: Técnicas para Maximizar o Valor do Projeto
A priorização do backlog é uma etapa fundamental no gerenciamento de projetos ágeis. Ela garante que as tarefas mais importantes e valiosas sejam realizadas primeiro, maximizando assim o retorno sobre o investimento (ROI) e a satisfação do cliente. Neste artigo, vamos explorar algumas das técnicas mais eficazes para priorização do backlog, incluindo Planning Poker, o método MoSCoW e outras abordagens úteis.

1. Método MoSCoW

O método MoSCoW é uma técnica de priorização simples, mas eficaz, que categoriza as tarefas em quatro grupos:

Must-Haves (Deve ter): Tarefas críticas sem as quais o projeto não pode ser considerado um sucesso.
Should-Haves (Deveria ter): Tarefas importantes, mas não essenciais para o sucesso do projeto.
Could-Haves (Poderia ter): Tarefas desejáveis, mas não necessárias.
Won't-Haves (Não terá): Tarefas que não são prioritárias ou que podem ser descartadas.
Vantagens: Fácil de entender e aplicar, ajuda a focar nas tarefas mais importantes.

2. Técnica de Eisenhower

A Técnica de Eisenhower, também conhecida como Matriz de Eisenhower, é uma ferramenta de priorização que ajuda a distinguir entre tarefas urgentes e importantes. Ela divide as tarefas em quatro quadrantes:

Urgente e Importante: Tarefas que precisam ser feitas imediatamente.
Não Urgente, mas Importante: Tarefas que são importantes para o projeto, mas não têm um prazo iminente.
Urgente, mas Não Importante: Tarefas que podem ser delegadas.
Não Urgente e Não Importante: Tarefas que podem ser eliminadas.

Vantagens: Ajuda a gerenciar o tempo de forma eficaz, focando em tarefas que realmente importam.

3. Kano Model

O Modelo Kano é uma técnica de priorização que classifica as características do produto em três categorias:

Básicas (Expected): Características que são esperadas pelo cliente.
Performance (More is Better): Características que melhoram a satisfação do cliente quanto mais são oferecidas.
Excitação (Surprise): Características inesperadas que proporcionam grande satisfação.

Vantagens: Ajuda a entender melhor as necessidades do cliente e a priorizar características que terão o maior impacto na satisfação do cliente.

4. Value vs Complexity

Esta técnica envolve plotar as tarefas em um gráfico com base em seu valor para o negócio e complexidade. As tarefas que oferecem alto valor com baixa complexidade são priorizadas primeiro.

Vantagens: Fácil de visualizar e ajuda a equipe a focar em tarefas que oferecem o melhor retorno sobre o investimento.

5-O Planning Poker não faz mais sentido

Com o advento do Kanban para Scrum(vide certificação PSK1), não se estima mais nada em story points, usa-se dados reais como Cycle Time e throughput.

Conclusão

A priorização do backlog é crucial para o sucesso de qualquer projeto. Utilizando técnicas como Planning Poker, método MoSCoW, Técnica de Eisenhower, Modelo Kano e Value vs Complexity, as equipes podem garantir que estão trabalhando nas tarefas mais importantes e valiosas. Cada técnica tem suas vantagens e pode ser escolhida com base nas necessidades específicas do projeto e da equipe. Ao priorizar de forma eficaz, as equipes podem maximizar o valor do projeto, melhorar a satisfação do cliente e alcançar os objetivos do negócio.

Lidando com membros tóxicos e panelinhas no time Scrum.

É fundamental compreender os indícios antes de agir:

Membros tóxicos: Atitudes como censuras frequentes sem apresentar alternativas, resistência de forma passivo-agressiva a inovações, ou rumores que comprometem a harmonia do grupo.

Panelinhas: Conjuntos que deliberam de maneira independente, deixam de lado outros membros durante conversas ou estabelecem uma dinâmica de "nós contra eles" dentro do grupo.

O Scrum Master precisa atentar para as interações que ocorrem em eventos como a Daily Scrum e a Sprint Retrospective. Por exemplo, se alguns participantes monopolizam as discussões enquanto outros ficam à margem, ou se escolhas são realizadas fora dos processos formais, é o momento de intervir.

1.Criando um Ambiente Seguro e Inclusivo.

Ter um ambiente onde as pessoas se sintam seguras emocionalmente é essencial para equipes saudáveis. O Scrum Master pode ajudar nisso de várias maneiras:

- Promovendo conversas abertas: Durante as retrospectivas, incentivar que todos possam falar sobre comportamentos e obstáculos sem medo de julgamento. Perguntas como "O que está dificultando nossa colaboração?" ajudam a estimular a reflexão de forma construtiva.

- Estabelecendo regras de convivência: Juntos, definir acordos simples, como respeitar todas as opiniões ou evitar conversas paralelas que possam prejudicar o foco do grupo(um trabalho árduo em tempos de home-office).

- Valorizando diferentes ideias: Garantir que todos tenham oportunidade de participar, usando técnicas como o round-robin, onde cada pessoa fala uma de cada vez, para que todas as vozes sejam ouvidas.

2.Lidando com Membros Tóxicos.

Conversa direta: Converse em particular com a pessoa, usando exemplos específicos, como: "Percebi que durante o planejamento, você interrompeu o João três vezes. Como podemos melhorar isso?" É importante evitar acusações e focar no impacto que o comportamento tem no time.

Oferecer ajuda: Às vezes, a toxicidade pode estar relacionada a frustrações que ainda não foram resolvidas.
Como o Scrum Master pode ajudar a remover obstáculos pessoais ou profissionais?

Estabelecendo limites claros: Na maioria das vezes foi possível melhorar a situação, mas, se o comportamento continuar, é importante envolver a liderança, como RH ou gestores, para tomar as ações necessárias, sempre seguindo as políticas da organização.

3.Como evitar conflitos internos.

Conflitos internos muitas vezes acontecem por falta de confiança ou por objetivos diferentes entre os membros da equipe. Para evitar essas situações, o Scrum master pode:

- Incentivar atividades que promovam a colaboração, como um revisar o código do outro(e assumirem o resultado do código em produção),

trabalho em grupo ou workshops que envolvam pessoas de diferentes áreas.

- Reforçar o objetivo comum: lembre a equipe da visão do produto e de como cada pessoa tem um papel importante para alcançar esse objetivo.

- Questionar decisões tomadas de forma isolada: se um subgrupo decidir algo sem consultar o restante do time, traga o assunto para uma conversa maior(preferencialmente na Retrospectiva da Sprint), perguntando: "Como podemos garantir que todos participem dessa discussão?"

4.Situações reais que vivenciei.

Em 15 anos passei por diversas situações envolvendo integrantes ácidos e panelinhas.
Seniores que estavam errados, mas defendiam o sempre foi assim.
Juniores que estavam errados e usavam o etarismo como forma de defesa.
Panelinhas envolvendo um “dominante” que usava sua influência para derrubar outro membro e assim se beneficiar da situação (salário).
Perseguições a membros por motivo de política.
Panelinhas visando derrubar o Scrum Master porque não queriam mudanças na cultura da equipe.
Membros que tinham mania de perseguição.
Membros que, por se sentirem frustrados,ou pouco valorizados na empresa, tentavam afundar o barco com todo mundo dentro.
Membros dos três papéis(Scrum masters,devs e P.O´s) que tinham deficiências técnicas mas não aceitavam nenhum tipo de crítica.

5. Prevenção: Cultivando uma Cultura Ágil Sólida.

A abordagem mais eficaz é antecipar e evitar dificuldades antes que elas apareçam.

-Aplique os princípios do Scrum: O Scrum Master deve exemplificar respeito, clareza e ousadia em suas atitudes.

-Aprecie as vitórias em grupo: Valorize a colaboração entre os membros, em vez de focar apenas nas realizações pessoais.

-Aposte na educação permanente: Cursos sobre a comunicação não violenta ou a administração de conflitos aprimoram a maturidade da equipe.

Conclusão:

Ter membros tóxicos e panelinhas na equipe representa desafios delicados, porém superáveis. O Scrum Master precisa atuar como um líder que apoia, combinando compreensão e determinação para recuperar a confiança e estimular a colaboração. Criando um espaço seguro para conversas abertas, promovendo diálogos sinceros e reforçando o propósito comum, é possível converter conflitos em oportunidades de desenvolvimento para o time. É importante lembrar que uma equipe coesa não é composta por ausência de divergências, mas sim pela capacidade de resolvê-las com respeito e foco na meta compartilhada.

Página deixada propositalmente em branco

As diversas configurações possíveis do time Scrum.

Um time Scrum deve ter entre 3 e 10 pessoas, obrigatoriamente deve ter um P.O, um Scrum master e no mínimo um desenvolvedor.

Um produto deve ter apenas um product owner.
Um Product owner pode estar em mais de um time e tomar conta de mais de um produto.

Um Scrum master pode estar em mais de um time.

Um Developer pode estar em mais de um time, apesar de muito raro.

Um Developer pode pular de equipe em equipe, a cada sprint(no caso de tecnologias legadas, com mão de obra difícil, por exemplo).

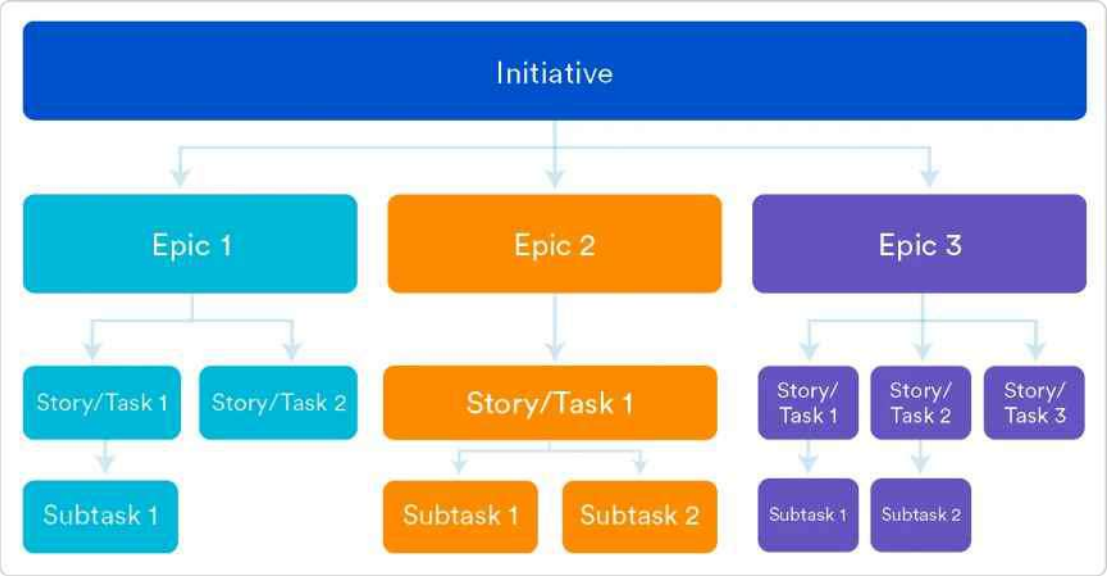
Um QA não é nada mais que um desenvolvedor que entende muito de testes. Na minha opinião, um QA deve ser mais um coordenador de testes do que um testador.

Conclusão:Pode, não quer dizer que seja o ideal, empresas têm tamanhos e necessidades diferentes, e o Scrum se adapta a todas elas.

Histórias de usuário,critérios de aceitação e tasks, tomando como exemplo a Java Pet Store.

No desenvolvimento de software, a capacidade de contar histórias de forma eficaz é fundamental para transformar requisitos vagos em soluções concretas. Inspirados nos ensinamentos da Scrum.org, vamos explorar como criar histórias e decompor essas histórias em tarefas gerenciáveis, utilizando o exemplo da Java Pet Store.
O que é Storytelling no Contexto de Desenvolvimento de Software?

No contexto do desenvolvimento de software, storytelling refere-se à prática de capturar requisitos ou funcionalidades desejadas do sistema em forma de histórias de usuário. Essas histórias são narrativas curtas que descrevem algo que o usuário deseja realizar dentro do sistema. Elas são fundamentais para o Agile e Scrum, pois ajudam a manter o foco no valor entregue ao usuário.



Criando Histórias Eficazes

Para criar histórias eficazes, devemos seguir alguns princípios básicos:

Histórias devem ser centradas no usuário: Elas devem descrever claramente quem é o usuário, o que ele deseja realizar e por quê.

Use o formato padrão: Utilize o formato "Como um [usuário], eu quero [realizar algo] para [atingir algum objetivo]".

Mantenha-as pequenas e gerenciáveis: Histórias devem ser granulares o suficiente para serem concluídas dentro de um sprint.

Exemplo: Java Pet Store
Vamos considerar o exemplo da Java Pet Store, uma aplicação de demonstração que simula uma loja de animais de estimação online. Uma história para essa aplicação poderia ser:
História:"Como um cliente da loja, eu quero visualizar os detalhes de um produto específico para decidir se devo comprá-lo."
Critérios de Aceitação: O cliente pode visualizar os detalhes do produto, incluindo nome, descrição, preço e imagem.
Os detalhes do produto são exibidos corretamente para diferentes tipos de produtos (cachorros, gatos, etc.).

Decompondo a História em Tarefas
Agora, vamos decompor essa história em tarefas técnicas que os desenvolvedores possam realizar:
Tarefa 1: Criar uma página de detalhes do produto

Descrição: Desenvolver a interface da página com os detalhes do produto.
Critérios de Aceitação:
A página deve exibir os detalhes do produto corretamente.
A página deve ser responsiva e adaptar-se a diferentes dispositivos.

Tarefa 2: Implementar a lógica de carregamento de detalhes do produto

Descrição: Implementar a lógica de negócios para carregar os detalhes do produto.
Critérios de Aceitação:
Os detalhes do produto são carregados corretamente do banco de dados.
A lógica de carregamento é eficiente e não causa problemas de desempenho.

Tarefa 3: Adicionar um carrossel de imagens

Descrição: Adicionar um carrossel de imagens para os produtos.
Critérios de Aceitação:
O carrossel de imagens é exibido corretamente na página de detalhes do produto.
O carrossel de imagens é responsivo e adaptar-se a diferentes dispositivos.

Tarefa 4: Testar a funcionalidade

Descrição: Escrever testes unitários e de integração para garantir que a funcionalidade esteja correta.

Critérios de Aceitação:

Os testes unitários e de integração são escritos e passam com sucesso.

A funcionalidade é testada e aprovada pelo Product Owner.

Conclusão

Contar histórias no desenvolvimento de software não é apenas uma técnica para capturar requisitos; é uma forma de garantir que o desenvolvimento esteja sempre focado no usuário e no valor que ele recebe. Ao seguir os princípios da Scrum.org e decompor histórias em tarefas gerenciáveis, como demonstrado com o exemplo da Java Pet Store, podemos garantir a entrega de software de alta qualidade que atende às necessidades dos nossos usuários.

Integrando Design Thinking ao Scrum.

Como Sessões Criativas Podem Aprimorar o Refinamento do Backlog.

No processo ágil de desenvolvimento de produtos, Scrum e Design Thinking são comumente considerados frameworks que se complementam. O Scrum fornece uma estrutura para entregas que ocorrem de forma iterativa e incremental, enquanto o Design Thinking adota uma perspectiva focada no usuário, incentivando a empatia, a geração de ideias e a criação rápida de protótipos.

Mas de que forma esses dois frameworks podem ser combinados na rotina de um time Scrum? Uma prática frequente é contar com um desenvolvedor focado em UX e Design Thinking, porém há um consenso crescente de que todos os desenvolvedores e o Product Owner (PO) devem aprimorar essas competências. Além disso, sessões de Design Thinking podem (e precisam!) acontecer durante a sprint, assim como o refinamento do backlog — e, em diversas situações, podem até ser vistas como parte do refinamento do backlog.

Por Que Todos no Time Devem Entender Design Thinking?
Ter um especialista em UX no time é valioso, mas limitar esse conhecimento a uma única pessoa pode criar um gargalo. Se todos os desenvolvedores e o PO tiverem noções de Design Thinking, o time ganha:

Melhor entendimento do usuário, resultando em soluções mais relevantes.

Colaboração mais eficiente entre desenvolvimento e design.

Backlog mais refinado, com histórias de usuário mais claras e validadas.

O PO, em particular, se beneficia ao incorporar técnicas de Design Thinking, pois pode priorizar demandas com base em evidências reais e não apenas em suposições.

Sessões de Design Thinking Durante a Sprint: O Scrum é um framework adaptável que permite reuniões além das cerimônias oficiais. Assim como o refinamento de backlog, uma prática recomendada, sessões de Design

Thinking também podem ser integradas ao fluxo da sprint de forma natural, enriquecendo o processo e os resultados do time.

É importante destacar que a presença dos stakeholders é sempre bem recebida em qualquer encontro de refinamento do backlog, especialmente quando se trata de sessões de design thinking,
Essas sessões podem incluir:

- Brainstormings rápidos para explorar soluções antes do desenvolvimento.
-Testes de usabilidade com usuários reais para validar hipóteses.
- Prototipagem low-fi para alinhar expectativas antes da codificação.
Essas Sessões Podem Ser Consideradas Refinamento?
Sim! Alguns especialistas argumentam que atividades de ideação e validação são parte do refinamento do backlog, pois ajudam a:

Clarificar requisitos antes de transformá-los em tarefas técnicas.

Validar suposições antes de comprometer esforço de desenvolvimento.

Alinhar expectativas entre PO, designers e desenvolvedores.

Jeff Patton, criador do User Story Mapping, defende que refinamento não é apenas decompor tarefas, mas entender profundamente o problema. Da mesma forma, **Jake Knapp**, autor do Design Sprint, mostra como ciclos curtos de ideação podem evitar retrabalho no desenvolvimento.

Conclusão: Design Thinking Não é um Processo Separado, Mas Parte do Fluxo Ágil
A integração entre Design Thinking e Scrum não só é possível, como deve ser incentivada. Em vez de tratar UX e desenvolvimento como etapas isoladas, times ágeis podem incorporar sessões criativas dentro da sprint, tratando-as como parte do refinamento contínuo do backlog.

Estratégias para que o QA não vire o gargalo da Sprint.

Primeiramente deve-se falar que o Scrum não reconhece diferença entre desenvolvedores, o QA é considerado como um desenvolvedor.
Há que se respeitar a disciplina de testes, ela é tão profunda quanto desenvolvimento de sistemas, mas o ideal é que cada desenvolvedor aprimore suas qualidades de testador, aprender sobre testes de integração e funcionais é um dever.

Se a equipe tiver um dev especializado em testes(leia-se QA), então esse deve funcionar como um conselheiro dos testes, deve ajudar no desenvolvimento do plano de testes, ele deve ratificar os testes feitos por cada dev, além de fazer os seus próprios, mas encontrar um erro nessa fase deve ser uma exceção, passível de ser discutida na retrospectiva da sprint.

QA não é babá de programador.

Nosso DEV entendido de testes pode ficar responsável por montar a pipeline de testes automatizados também.
O Scrum é um framework para desenvolvimento de produtos, não só software, um desenvolvedor pode ser um pintor de parede, todos os seus

devs programam em Java, ou em Oracle? Não, então o QA é um dev e pronto.
Um dev testar o trabalho do outro, antes de passar para o nosso dev especialista em testes(QA), costuma dar bons resultados.
Os testes devem fazer parte da definition of done e da definition of workflow, evitando-se sobrecarregar o QA e deixar tudo para a última hora.
Um bom Scrum Master deve saber sobre WIP, definição de workflow e Service level expectation para ajudar o time em suas definições.
A empresa pagar um curso de testes na Udemy é uma boa estratégia.

A certificação DEV da Scrum.org aborda muito a área de testes.
O Scrum é um framework adaptável a realidade e as práticas da empresa(isso não deve ser confundido com Scrum relaxado).
Quanto mais o P.O entende do produto, e é capaz de transformar isto em boas histórias, menos precisa-se dos stakeholders para testes de UAT, mas não há nada que impeça que os stakeholders possam dar uma forcinha com os testes, isso é especialmente útil quando a equipe trabalha com sistemas legados e complexos.
É muito importante que nosso dev entendido de testes participe ativamente do refinamento do backlog.

Muito embora a review não deva ser vista como um portal de envio de valor para a produção, espera-se que a maioria do que foi combinado seja demonstrado ao fim da sprint, não vale deixar tudo para o QA conferir na quinta-feira véspera da review, quando muitas vezes é tarde para se arrumar alguma coisa.
Defeito não é débito técnico, é defeito mesmo, se algo está defeituoso, não bate com a definition of done, e não deve ser demonstrado na review.
Apesar da entrega contínua diminuir a pressão da entrega na review, não se deve fazê-lo por falhas no processo

Conclusão: QA não é babá de desenvolvedor, todos da equipe Scrum devem entender de testes, a função do QA é coordenar e garantir a qualidade dos testes, ele deve meter a mão na massa também, mas de maneira lógica e coordenada.

CAMPANHA QA NÃO É BABÁ.

Substituição no time Scrum: Saem Story points e Capacity, entram Throughput e Cycle time.

Desde 2012, Ken Schwaber(provavelmente inspirado no Lean Kanban), um dos criadores do Scrum, tem questionado a validade do uso da métrica como velocity (velocidade) dentro do framework Scrum.

Em seu artigo "Waste Not", ele argumenta que gastar tempo se preocupando com essa métrica é um desperdício que não traz valor real ao processo de desenvolvimento.

Essa opinião fica ainda mais importante quando pensamos em equipes Scrum que também usam práticas de Kanban, onde indicadores como throughput(taxa de entrega) e cycle time (tempo de ciclo) têm se mostrado muito mais eficientes para planejar o trabalho que será incluído no Sprint Backlog.

Ron Jeffries, o criador dos story points, já não os utiliza mais. Caso tenha interesse, procure por Ron Jeffries "Story Points-Sorry". Infelizmente, as previsões baseadas em story points tornaram-se uma espécie de muleta que, por um longo período, ajudou a aliviar a pressão sobre as equipes Scrum. No entanto, agora está evidente que essas estimativas trazem mais prejuízos do que benefícios. Ainda assim, é

difícil convencer muitos Scrum Masters, especialmente aqueles que não possuem certificação pela Scrum.org, a abandoná-las em favor da adoção de dados reais, como cycle time e throughput.

No contexto atual, elas são amplamente vistas como desnecessárias e, sob a ótica do Lean, podem ser consideradas desperdício. Consigo até imaginar a reação de choque de alguns Scrum Masters ao ler isso.

Os Problemas Fundamentais com Velocity

Velocity: Uma Métrica Enganosa
O velocity, tradicionalmente medido em story points por Sprint, apresenta vários problemas:

Falsa precisão: Story points são estimativas subjetivas que variam entre times e até mesmo entre membros do mesmo time ao longo do tempo .

Comparações inválidas: Como Ken Schwaber apontou, velocity não deve ser usado para comparar times, pois é uma métrica específica de cada equipe.

Foco errado: Times podem começar a otimizar localmente para aumentar velocity, em detrimento da entrega de valor real ao cliente.

A nova abordagem

A Abordagem Scrum com Kanban: throughput e Cycle Time
A certificação Professional Scrum with Kanban (PSK), oferecida pela Scrum.org, apresenta uma abordagem diferente, que se baseia nos princípios de fluxo do Kanban. Nesse método, duas métricas principais se destacam:

Throughput: Quantidade de Itens Entregues
Definido como "o número de itens de trabalho concluídos por unidade de tempo" , o throughput oferece:

Medição objetiva: Baseada em itens reais entregues, não em estimativas.

Maior precisão: Padrões de troughput históricos permitem previsões um pouco mais acuradas(lembre-se não existe algo como bola de cristal).

Simplicidade: Fácil de medir e entender por todas as partes interessadas.

Cycle Time: Tempo para Concluir um Item
O cycle time mede quanto tempo leva para um item fluir do início ao fim no workflow da equipe. Suas vantagens incluem:

Identificação de gargalos: Mostra onde o trabalho está entalando.

Previsão de entrega: Permite estimar quando itens futuros serão concluídos

Melhoria contínua: Reduzir cycle time é um objetivo claro de melhoria de processo.

Service Level Expectation (SLE): O Novo Balizador para Sprint Planning

1.Um conceito importante que aprendemos na preparação para a certificação PSK é o Service Level Expectation, ou SLE. Ele é basicamente uma previsão de quanto tempo deve levar para uma determinada tarefa passar por todo o processo dentro do fluxo de trabalho da equipe Scrum.

Na prática, o SLE funciona como uma alternativa às estimativas tradicionais durante o planejamento do Sprint.
Decisão rápida: Em vez de longas sessões de estimativa, a equipe simplesmente pergunta: Este PBI pode ser concluído dentro do nosso SLE?

Foco no tamanho certo: Itens muito grandes para o SLE são divididos antes de entrarem no Sprint Backlog(decomposição das histórias em tasks).

Adaptação contínua: O SLE é refinado com base em dados históricos reais de cycle time

Implementando throughput-Driven Sprint Planning
Baseado nos materiais da Scrum.org para a certificação PSK, aqui está como um time pode implementar o planejamento baseado em throughput:

Visualize o fluxo: Crie um Kanban board que mostre claramente os estágios do workflow.(procure por definition of Workflow Scrum.org).

Limite WIP: Estabeleça limites de trabalho em progresso para cada estágio.

Meça throughput histórico: Analise quantos itens foram concluídos nos últimos Sprints.

Defina SLE: Estabeleça uma expectativa realista baseada em cycle times históricos.

Planeje com throughput: No Sprint Planning, selecione itens até atingir o throughput médio histórico.

Times que adotam esta abordagem costumam dizer que:

Maior previsibilidade: Throughput é mais estável que velocity em story points.

Menos tempo gasto em estimativas: Decisões baseadas em SLE são feitas em segundos.

Melhor fluxo de valor: Limites de WIP impedem sobrecarga e melhoram a qualidade.

Transparência: Métricas baseadas em dados reais criam confiança com stakeholders.

Conclusão: Ken Schwaber, já em 2012, apontava que a métrica como velocity poderia ser onsiderada como desperdício em processos ágeis. A abordagem apresentada na certificação PSK da Scrum.org, com ênfase em throughput e cycle time, surge como uma alternativa mais alinhada aos princípios do empirismo e da melhoria contínua característicos do pensamento ágil. Para equipes que já dominam os fundamentos do Scrum, incorporar práticas do Kanban pode representar um avanço natural em sua jornada ágil. Essa adoção promove maior previsibilidade, otimização do fluxo de trabalho e, acima de tudo, entrega consistente de valor ao cliente. As equipes interessadas em explorar mais profundamente essa abordagem têm à disposição os recursos oferecidos pela Scrum.org para a certificação Professional Scrum with Kanban. Essa certificação fornece um guia claro para integrar essas práticas mantendo a essência dos princípios fundamentais do Scrum.

Wip, Cycle time, Lead time, Throughput , Lei de Little.....

Esse é o único artigo chato desse livro, esse artigo não tema pretensão de preparar ninguém para o PSK1, apenas dar uma pincelada sobre o assunto, e instigar a curiosiade do leitor.

Estou convencido de que, assim que você assimilar tudo, encontrará muito mais facilidade e agilidade ao lidar com esses indicadores do que permanecer apenas no universo da imaginação, refletindo sobre story points, capacidade e velocidade.

Na **PSK1** (Professional Scrum Kanban Level 1), **Leading** e **Lag Indicators** são métricas usadas para monitorar e melhorar o fluxo de trabalho no contexto de Scrum combinado com Kanban. Vamos entender cada um e sua aplicação nas cerimônias do Scrum:

Quando usamos o Scrum junto com o Kanban, os indicadores chamados leading, ou antecipatórios, ajudam a prever o que pode acontecer no futuro. Já os indicadores lagging, ou atrasados, mostram o que já ocorreu. Entre os exemplos de indicadores leading estão o "Work in Progress" (WIP) e a "Work item age". Já os indicadores lagging incluem o "Cycle Time" (tempo de ciclo) e a "Throughput" (produtividade).

1. Conceitos Fundamentais

(A) Cycle Time vs. Lead Time

Definições:

Cycle Time: Tempo ativo para concluir um item (ex: de "In Progress" a "Done")4 dias (codificação + teste)

L**ead Time:**Tempo total desde a solicitação até a entrega10 dias (incluindo espera no backlog)

Work Item Age:(idade do item de trabalho) refere-se à quantidade de tempo que um item de trabalho (tarefa, bug, etc.) permanece em um estado de fluxo no quadro Kanban

Aplicação no Scrum com Kanban:

Lead Time é essencial para compreender a rapidez na entrega de valor. Avalia o período total desde a elaboração de uma tarefa até a sua finalização, oferecendo dados significativos sobre a eficácia do fluxo de trabalho.

Cycle Time ajuda a equipe a otimizar o fluxo,pois:Ajuda na medição da eficiência, identificação de gargalos, otimização de processos e ajuda na definição de prazos .

(B) Throughput (Taxa de Entrega)

Definição: Número de itens concluídos por unidade de tempo (ex: 5 itens/sprint).
Fórmula:
Throughput = Itens concluídos / Tempo (ex: por sprint)

(C) Lei de Little: A Base Matemática do Fluxo
WIP = Throughput × Cycle Time
Exemplo:

Se uma equipe tem Throughput = 10 itens/sprint e Cycle Time = 2 dias/item, então:

WIP ideal = 10 × 2 = 20 "dias de trabalho" em progresso

Assim sendo limita-se p WIP da seguinte forma: Se a sprint tem 10 dias úteis, o WIP máximo deve ser 2 itens em paralelo (20 dias / 10 dias).

2. Definição de Workflow e WIP no Scrum com Kanban
(A) Workflow Básico (Exemplo de colunas Kanban)
Backlog → Refinado → Em Desenvolvimento → Em Teste → Pronto para Deploy → Done

Políticas importantes:

Definição de Pronto (DoD) clara para cada etapa.
Limites de WIP para evitar sobrecarga.

(B) Como Calcular WIP no Início da Sprint?

Fatores que impactam:
-Feriados: Reduzem dias úteis.
-Férias: Diminuem capacidade da equipe.
-Exemplo Prático:
Equipe: 5 devs + 1 QA(eu sei que le é um dev) = 6 pessoas

Sprint: 2 semanas (10 dias úteis), mas há 2 feriados e 1 dev de férias.

Dias úteis ajustados:

Dias totais = 10 - 2 feriados = 8 dias
Capacidade = 4 devs (pois 1 está de férias) + 1 QA = 5 pessoas
Capacidade total = 8 dias × 5 pessoas = 40 "dias de trabalho"

Se o Cycle Time médio é 5 dias/item:

WIP máximo = Capacidade / Cycle Time = 40 / 5 = 8 itens

-Limites sugeridos:

Desenvolvimento: 4 itens (1 por dev)
Teste: 2 itens (evitar gargalo no QA)-Aí vê-se a importância dos desenvolvedores aprenderem sobre estes.

3. Cenário Crítico: QA como Gargalo (e Seu Impacto no Fluxo)
(A) O Problema

Time com apenas 1 QA para 4 devs.

Workflow sem políticas claras: Devs "empurram" itens para teste sem verificação prévia.
Resultado:
Fila enorme em "Em Teste" (WIP explode).
Cycle Time aumenta (Lei de Little em ação).

(C) Soluções Baseadas no PSK I
Limitar WIP em "Em Teste" (ex: máximo 2 itens).

Incluir testes no Definition of Done dos devs (ex: "Nenhum item vai para teste sem testes unitários").
Cross-training: Devs devem aprender testes básicos.

4. Erro Comum: O problema dos devs que não testam e pensam que o Q.A é sua babá.

Viola o princípio "Done = Potencialmente Entregável" do Scrum.
Cria dependência excessiva em QA.
Efeito no WIP: Itens ficam "presos" em teste, aumentando Cycle Time.

Como corrigir:

Incluir testes no workflow dos devs:
Em Desenvolvimento → Testes Unitários/Integração → Revisão → Em Teste (QA) → Done

Definir políticas claras:
"Nenhum item vai para QA sem 90% de cobertura de testes".
Usar pair testing (dev + QA).

5. Ferramentas para Visualizar e Melhorar o Fluxo

Cumulative flow diagram. Sua principal função é representar graficamente o progresso das atividades, permitindo uma visualização clara do andamento do trabalho e facilitando a identificação de possíveis gargalos no processo. Ele possibilita monitorar o andamento das tarefas ao longo do tempo, exibindo a quantidade de itens em cada fase do processo, desde o início até a finalização.

WIP: Work in progress

Gargalos: Colunas que empacam, não podem receber mais itens,pois tem itens parados,e ofenderiam a regra de limitação do WIP (ex: "Em Teste").

(B) Cycle Time- **Control Chart**

(C) Aging Work in Progress
Mostra itens "envelhecendo" no fluxo (ex: um bug parado há 3 dias em teste).Pode ser visualizado através do **Aging WIP Chart.**

Para uma análise mais detalhada, o melhor artigo(em inglês) que conheço e que ilustra cada um desses gráficos, assim como suas aplicações nas diversas cerimônias, é o seguinte:

https://www.scrum.org/resources/blog/4-key-flow-metrics-and-how-use-them-scrums-events

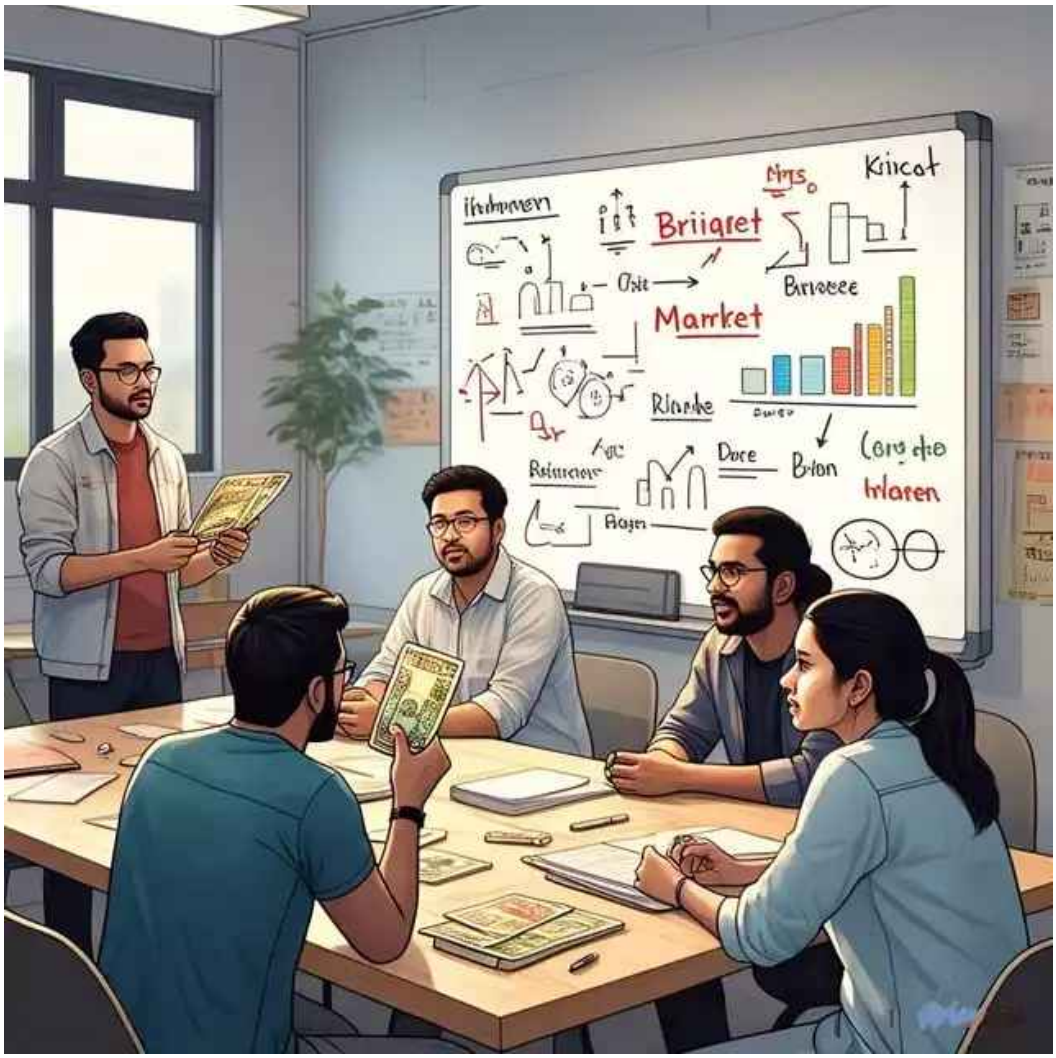
O livro(em inglês) que me ajudou, em muito, a tirar a PSK1 foi: https://www.amazon.co.uk/Applying-Scrum-Kanban-Pointless-Book/dp/B09HFZCJFT

Na minha opinião, a prova da PSK1 tem um nível de dificuldade bem maior que a PSM1, tornando a leitura do livro acima obrigatória.

Dinâmicas visando fortalecer os valores do Scrum.

Os Valores Scrum são: Comprometimento, Coragem, Foco, Abertura e Respeito

O Scrum Master ocupa uma posição crucial na defesa dos princípios do Scrum: Compromisso, Coragem, Foco, Abertura e Respeito. Esses princípios são vitais para equipes ágeis com alto desempenho, mas exigem um cuidado deliberado para que sejam mantidos. Quando ignorados, podem resultar em desavenças, queda na produtividade e falta de alinhamento. Neste texto, abordaremos estratégias práticas que o Scrum Master pode implementar para reforçar cada um desses valores, promovendo uma mudança positiva na cultura da equipe.



1. Comprometimento (Commitment)

O Comprometimento diz respeito à entrega da equipe às objetivos comuns e à excelência do serviço, respeitando os entendimentos firmados.

"Mapa do Compromisso": No início da Sprint, o time desenha um "mapa" com os objetivos da Sprint no centro. Cada membro escreve em post-its: "Como posso contribuir?" e "O que preciso dos outros?". Todos compartilham respostas e negociam apoios. Isso torna explícitas as expectativas individuais e coletivas, criando um pacto visual de responsabilidades.

"Check-in da Responsabilidade": Diariamente, após o Daily Scrum, reserve 5 minutos para uma rodada rápida: "O que prometi ontem e cumpri? O que prometo hoje?". Registre os compromissos em um quadro compartilhado. A transparência constante reforça a confiança mútua e a cultura de accountability.

"Contrato de Comprometimento": Crie um contrato que todos os membros do time assinam, comprometendo-se a trabalhar em equipe e a respeitar os acordos estabelecidos.

2. Coragem (Courage)

A coragem envolve a capacidade de enfrentar desafios difíceis, dar feedback honesto e experimentar novos caminhos.

"O Elefante na Sala": Em uma reunião de feedback, distribua papéis anônimos. Cada pessoa escreve um "elefante" (problema não dito). Os papéis são sorteados e discutidos pelo grupo, focando em soluções. O anonimato inicial reduz o medo de represálias, encorajando verdades difíceis.

"Desafio do Experimentador": O time elege um "experimento ousado" por Sprint. No final, discutem: "O que aprendemos, mesmo falhando?". Isso normaliza o risco e destaca que coragem é sobre aprendizagem, não perfeição.

"Mestre do Caos": Designe um membro do time para ser o "Mestre do Caos" em cada Sprint. Essa pessoa é responsável por identificar e resolver problemas críticos, promovendo a coragem e a proatividade.

3. Foco (Focus)

O foco envolve priorizar atividades que agregam valor direto à meta da Sprint, minimizando distrações.

"Caça às Distrações": Em uma reunião, liste todas as atividades paralelas do time. Classifiquem-nas em "Essenciais" ou "Ruído". Negociem a eliminação de pelo menos "ruídos". Isso empodera o time a proteger seu tempo coletivo.

"Pomodoro Coletivo": Em dias de alta carga, o time trabalha em blocos de 25 minutos sincronizados. Após cada bloco, uma pausa rápida compartilhada. O ritmo sincronizado reduz multitasking e cria fluxo concentrado.

"Quadro de Prioridades": Crie um quadro que liste as prioridades do time. Isso ajuda a manter o foco nas atividades mais importantes.

4. Abertura (Openness)

A abertura envolve transparência sobre progressos, obstáculos e ideias, criando um espaço seguro para vulnerabilidade.

"Show & Tell de Erros": Na Retrospectiva, cada membro compartilha um erro cometido na Sprint e como o corrigiu. Celebra-se o aprendizado, não a culpa. Isso desmistifica o fracasso e incentiva a inovação.

"Mercado de Ideias": Crie um "mercado" virtual onde todos colam ideias para melhorias. Cada pessoa "investe" pontos em 3 ideias alheias. As mais votadas viram ações. Isso democratiza a criatividade e mostra que todas as vozes importam.

"Sessão de Feedback": Reserve um tempo para que os membros do time compartilhem feedback uns dos outros. Isso promove a abertura e a transparência.

5. Respeito (Respect)

O respeito envolve valorizar as contribuições, habilidades e limites de cada membro, promovendo empatia.

"Presente de Feedback": Ao final da Sprint, cada pessoa escreve para outro membro: "Um talento seu que admirei foi..." e "Uma ideia para

você explorar é...". Compartilhem em voz alta. Isso constrói reconhecimento mútuo e críticas respeitosas.

"Persona do Colega": Em duplas, cada pessoa entrevista o parceiro sobre: "Como prefere receber feedback?", "O que te desmotiva?". Depois, apresentam um ao outro para o time. Isso promove compreensão das individualidades e evita conflitos.

"Agradecimentos": Reserve um tempo para que os membros do time agradeçam uns aos outros por suas contribuições. Isso promove o respeito e a gratidão.

Conclusão

Os princípios do Scrum são capacidades que precisam ser desenvolvidas. O Scrum Master deve funcionar como um "jardineiro da cultura ágil", utilizando dinâmicas desse tipo para cultivar sementes de confiança e trabalho em equipe. O principal sinal de sucesso não é a rapidez, mas sim a unidade do grupo. Inicie com uma dinâmica centrada em valores, ajuste-as de acordo com a realidade da equipe e veja como a maturidade ágil se desenvolve.

Teremos que cancelar a sprint , e agora?

O Scrum Guide é claro ao afirmar que o cancelamento só deve ocorrer quando o objetivo da Sprint se torna obsoleto . Mas o que exatamente torna um objetivo obsoleto? Na prática, diversos fatores externos e internos podem justificar essa decisão:

1. Mudanças Econômicas e de Mercado

Flutuações significativas na economia ou mudanças abruptas nas condições de mercado podem tornar o trabalho planejado obsoleto. Por exemplo:

Uma crise econômica que obriga a empresa a repriorizar iniciativas.
Mudanças nas taxas de juros que afetam a viabilidade de certos produtos.
Flutuações cambiais.
Novas taxas de importação e exportação.

2. Transformações Corporativas

Aquisições, fusões ou vendas de empresas podem alterar completamente as prioridades estratégicas:

Aquisição de uma subsidiária que traz novas capacidades e torna redundante o trabalho atual.

Venda de parte do negócio que torna o produto em desenvolvimento desnecessário.

Fusão com outra empresa que já possui solução similar.

3. Perda de Clientes Estratégicos

Quando um cliente chave desiste de um projeto.

Cancelamento de contrato com um cliente que representava parte significativa da receita.

Mudança nas exigências regulatórias que invalidam o trabalho planejado.

Concorrência que lança solução superior, ou com custo muito baixo, tornando o produto atual obsoleto.

4. Mudanças Tecnológicas.

Descobertas técnicas ou limitações que surgem durante a Sprint:

Tecnologia base se torna obsoleta ou insegura.

Descoberta de que a solução proposta não é tecnicamente viável.

Surgimento de nova tecnologia que oferece melhor custo-benefício.

5. Mudanças Legais ou Regulatórias.

Novas leis ou regulamentos que impóssibilitem o produto em desenvolvimento de ser lançado.

Jurisprudências que impactem a viabilidade do projeto.

Mudanças em padrões de compliance.

O Processo de Cancelamento: Papéis e Responsabilidades
É crucial entender que, no Scrum, apenas o Product Owner tem autoridade para cancelar uma Sprint . No entanto, essa decisão normalmente é tomada após consulta com a equipe e stakeholders importantes.

O que o Product Owner pode fazer?

Caso opte por cancelar a Sprint, o product owner deve informar e esclarecer as razões para os envolvidos (stakeholders, acionistas etc.) e para a equipe Scrum. Além disso, é importante que o product owner se dedique a organizar e reunir os recursos necessários para reordenar o backlog, com foco na próxima sprint.

O que o Scrum Master pode fazer?

O Scrum Master não decide, mas deve aconselhar,facilitar a discussão sobre o motivo da obsolescência do objetivo e ajudar a equipe a entender o motivo e as implicações do cancelamento.

O que os desenvolvedores podem fazer?

A equipe deve fornecer informações técnicas sobre o estado atual do trabalho,participar ativamente do replanejamento .

O Que Fazer Após o Cancelamento?
Quando uma Sprint é cancelada, é essencial agir rapidamente para minimizar os impactos negativos. A recomendação é iniciar

imediatamente uma nova Sprint, para retomar o ritmo, e readequar o calendário.

1. Planejar uma Sprint tampão

Uma abordagem eficaz é:

Criar uma Sprint de que termine em uma sexta-feira(o tamanho é indiferente, desde que não seja superior há um mês),não importa que uma Sprint começe em uma quarta,e não em uma segunda como de costume.

Focar em itens críticos que ajudem a realinhar o produto.

Terminar na sexta-feira permite que a equipe retome o ciclo normal na segunda-feira seguinte.

2. Lidar com o Trabalho Existente

Itens completos e "Prontos", ou seja, aqueles que satisfaçam a Definition of Done, devem ser revisados e aceitos, se ainda fizerem sentido.

Trabalho incompleto deve retornar ao Product Backlog(Se ainda fizer sentido).

3.Review se fizer sentido

Caso existam itens que tenham sido finalizados e que ainda façam sentido eles devem ser mostrados na review, do contrário será apenas desgaste emocional e perda de tempo dos envolvidos.

4. Realizar uma retrospectiva especial

Uma retrospectiva focada no cancelamento(no dia do cancelamento,ou no próximo) ajuda a:

Identificar lições aprendidas.

Verificar se a equipe teve alguma responsabilidade no evento, não visando punir ninguém, mas fechar as portas para que isso volte a acontecer.

5-Impactos e riscos do cancelamento

Cancelar uma Sprint não é uma decisão trivial. Como destaca Julio Roche da Deloitte, "as cancelamentos de Sprint consomem recursos e são frequentemente traumáticas para o Time Scrum"

5.1-Impacto na Moral da Equipe.

Frustração por trabalho aparentemente desperdiçado.
Sensação de instabilidade e incerteza.
Potencial diminuição da motivação.

5.2-Impacto Operacional.

Tempo perdido com replanejamento
Custo de interromper e reiniciar o trabalho
Potencial acúmulo de débito técnico.

5.3. Impacto na próxima Sprint.

O impacto na capacidade de selecionar itens para a próxima Sprint(Sprint Backlog) será diferente para cada nível de maturidade(equipes mais maduras serão menos afetadas), equipes que ainda usam story points(o que não é adequado) também sofrerão mais que equipes que se baseiam em cycle time e Troughput.

6-Melhores Práticas para Minimizar os Impactos Para reduzir os efeitos negativos do cancelamento:

6.1. Comunicação Clara e Transparente.

Explicar a todos os stakeholders os motivos da decisão.
Deixar claro que é uma medida excepcional.
Reafirmar o compromisso com os novos objetivos.

6.2. Replanejamento Ágil.

Realizar um novo Sprint Planning o mais rápido possível.
Manter a equipe focada no novo objetivo.
Estabelecer metas claras e alcançáveis .

6.3. Aprendizado Contínuo.

Usar a experiência para melhorar a resiliência.
Documentar lições aprendidas.
Ajustar processos para evitar situações similares.

Conclusão:

O cancelamento de uma Sprint sempre trás prejuízos financeiros e a moral, mas, não deve ser visto como algo absurdo, e sim como aprendizado, e as vezes, como risco do negócio.
O importante é focar nas medidas de contensão e realinhamento.
Nada substitui o empirismo, só dei algumas sugestões baseadas na minha vivência, mas sua equipe pode, e deve, ter ideias de como lidar com uma situação tão delicada como o cancelamento de uma Sprint.

Página deixada propositalmente em branco.

Quando o QA vira Product Owner.

Primeiro não é melhor ou pior ser QA,ou PO, estou escrevendo esse artigo por cansar de responder essa pergunta,para QA´s,gerentes e pessoal do RH.

A principal missão de um QA é garantir a qualidade do software. Essa responsabilidade exige um olhar crítico e meticuloso, capaz de identificar falhas funcionais, inconsistências e oportunidades de melhoria. Tal atenção aos detalhes é uma vantagem significativa no papel de PO, especialmente durante o refinamento de backlog, validação de requisitos e elaboração de critérios de aceite. Além disso, sua vivência com testes exploratórios os ensina a "pensar fora da caixa" — uma habilidade crucial para antecipar riscos e prever o comportamento real do usuário.
É preciso que se diga que conheço um QA que se arrependeu da mudança e voltou a ser QA, por uma questão de vocação mesmo.
É sempre bom lembrar que existe um plano de carreira para QA´s.

1. Proximidade com o Ciclo de Desenvolvimento

QAs estão inseridos no dia a dia dos times ágeis, acompanhando desde as daily meetings até as entregas de sprint. Eles entendem profundamente os ritos, artefatos e papéis do Scrum, o que facilita a adaptação às rotinas e responsabilidades do PO. Essa convivência constante com desenvolvedores e Scrum Masters ajuda na criação de uma ponte natural entre as áreas técnica e de negócio — papel central do PO.

2. Foco no Usuário Final e Experiência Real

Testar um produto não é apenas validar requisitos: é imaginar como o usuário vai interagir com ele. QAs bem-sucedidos desenvolvem empatia pelo cliente e são capazes de detectar fricções na experiência antes mesmo de chegarem ao mercado. Essa mentalidade centrada no usuário é essencial para qualquer PO que queira entregar valor contínuo.

3. Mentalidade de Prevenção e Melhoria Contínua

QAs vivem a cultura de melhoria contínua, através de retros, automações de teste, e feedbacks constantes. Essa visão crítica e proativa ajuda na evolução do produto e no aprimoramento de processos internos — características que agregam muito ao PO na priorização de features e refinamento de requisitos.

Desafios na Transição: O Que Ainda Falta aos QAs?

Apesar das vantagens, QAs que desejam migrar para o papel de PO enfrentam uma curva de aprendizagem importante. As principais lacunas estão no campo estratégico, comportamental e de liderança.

1. Visão Estratégica e Orientação a Valor

Enquanto QAs se concentram em encontrar erros e garantir estabilidade, o PO precisa definir o que construir com base em metas de negócio, necessidades do mercado e comportamento do cliente. Isso requer uma mudança de foco: de eficiência operacional para impacto estratégico. Muitos QAs ainda têm dificuldade em enxergar o produto como parte de um ecossistema maior, onde decisões precisam gerar valor mensurável.

2. Gestão de Stakeholders e Negociação de Interesses

O PO atua como elo entre áreas técnicas e executivas. É preciso lidar com múltiplos interesses — clientes, liderança, marketing, vendas,

suporte — e saber dizer “não” com base em dados e prioridades. Essa capacidade de influenciar e articular visões é pouco explorada por QAs em suas funções tradicionais, mas fundamental no novo papel.

3. Métricas de Produto e Negócio

Enquanto QAs se apoiam em KPIs como cobertura de testes, taxa de falhas ou bugs por release, o PO precisa acompanhar métricas como retenção, engajamento, churn, LTV e NPS. Mudar o mindset de "produto funcionando corretamente" para "produto gerando resultado" é um divisor de águas.

4. Tomada de Decisão em Ambientes Ambíguos

POs precisam priorizar backlog mesmo quando há incertezas, pressões externas e recursos limitados. A habilidade de tomar decisões com base em trade-offs — e assumir riscos calculados — não costuma ser natural para QAs, mas é indispensável para quem lidera a estratégia do produto.

Habilidades que um QA Deve Desenvolver para se Tornar um PO

A transição de QA para PO exige um esforço intencional de capacitação. Algumas habilidades essenciais incluem:

- **Business Acumen**

Compreender como o produto gera receita, qual o modelo de negócio, quem são os concorrentes e quais são as forças de mercado. Leitura de relatórios de mercado, cursos de análise de negócios e até um mini MBA são caminhos eficazes.

- **Priorização com Base em Valor**

Dominar frameworks como MoSCoW, RICE, WSJF e Kano Model para priorizar funcionalidades com base em impacto no negócio, esforço técnico e retorno para o cliente.

- **Storytelling e Comunicação Assertiva**

O PO precisa ser capaz de vender ideias, engajar a equipe e apresentar resultados para diferentes públicos. Desenvolver soft skills como oratória, escuta ativa e escrita estratégica é crucial.

- **Pesquisa com Usuários e Validação de Hipóteses**

Participar de entrevistas, testes de usabilidade, análise de mapas de calor e testes A/B. Conhecer ferramentas como Hotjar, Google Analytics e Amplitude pode ser um diferencial.

- **Gestão de Métricas de Produto**

Aprender a trabalhar com métricas de ativação, retenção, CAC, LTV e taxa de conversão. Mais do que números, o PO deve entender o "porquê" por trás dos dados e usá-los para guiar decisões.

Certificações que Impulsionam a Transição
Certificações não são pré-requisitos, mas agregam valor e demonstram compromisso com o novo papel. Algumas recomendadas:
Certified Scrum Product Owner **(CSPO)**
Professional Scrum Product Owner **(PSPO I e II)** – Aprofunda práticas e métricas de produto.

Conclusão: Muitos QA's são candidatos naturais a PO, mas essa transição deve ser feita sem correria, adquirindo os hard e soft skills necessários para a nova função, sem forçação de barra, de uma maneira consciente e com propósito.

Scrum para desenvolvimento e Kanban para sustentação.

A indústria de software evoluiu para um cenário onde a adaptabilidade e a entrega de valor contínuo são essenciais.

Nesse contexto, **Scrum** e **Kanban** emergem não como rivais, mas como abordagens complementares, cada uma com sua força: o primeiro para projetos de desenvolvimento, o segundo para a dinâmica imprevisível da sustentação. Este artigo explora como esses frameworks coexistem harmoniosamente, adaptando-se às necessidades modernas.

1-Scrum Evoluído: Flexibilidade e inovação contínua

O Scrum, tradicionalmente associado a sprints fixos e entregas ao final de cada ciclo, passou por transformações significativas. Hoje, ele incorpora práticas que o tornam mais ágil e adaptável:

-Entrega Contínua Desvinculada da Revisão de Sprint
A adoção de Continuous Deployment (CD) permite que incrementos sejam liberados a qualquer momento, desde que atendam à Definition of Done. Isso elimina a dependência exclusiva da Sprint Review para

entregas, acelerando o time-to-market sem comprometer a estrutura do framework.

A Review serve para se revisar o que foi feito durante a Sprint, se fosse para entregar algo ela se se chamaria Delivery meeting.

-Mudanças no Sprint Backlog
Embora o Sprint Goal permaneça imutável, o Sprint Backlog pode ser ajustado durante o ciclo, desde que negociado com o Product Owner (PO) e não comprometa o objetivo principal. Por exemplo, correções críticas ou demandas emergenciais podem ser incorporadas, garantindo flexibilidade sem desestabilizar o planejamento(ok, ok Se sua empresa tem pouquíssima sustentação, a ponto de não prejudicar a Sprint, o Scrum esta de bom tamanho).

-Acúmulo de Papéis com Responsabilidade
Em equipes menores ou multidisciplinares, é possível acumular funções (como um Scrum Master atuando também como desenvolvedor), desde que haja transparência e equilíbrio para evitar conflitos de interesse. Essa prática reflete a maturidade das equipes em ambientes ágeis.

Adoção de Práticas Lean e Kanban
Nos últimos anos, o Scrum incorporou princípios do Lean Kanban, como a limitação de WIP (Work in Progress), para reduzir gargalos e melhorar o fluxo de trabalho. Essa integração reforça a convergência entre os frameworks, já que ambos compartilham valores como transparência e melhoria contínua .

2-Kanban na sustentação: Simplicidade e resposta imediata

Enquanto o Scrum se adapta a projetos complexos, o Kanban brilha em ambientes de sustentação, onde a imprevisibilidade absoluta, o pouco conhecimento dos usuários finais, e o senso de urgência reinam:

Incidentes em produção não esperam por cerimônias.

Problemas críticos exigem ação imediata, sem aguardar planning ou priorização em sprints. O Kanban, com seu fluxo contínuo e quadros visuais, permite que as equipes "puxem" tarefas urgentes diretamente para execução, resolvendo falhas em tempo real.

Ausência de cerimônias com datas específicas
Sem eventos fixos como Daily Scrum ou Sprint Review, o Kanban oferece uma estrutura maleável, ideal para times de suporte que lidam com

demandas variáveis. A priorização dinâmica ajusta-se à criticidade dos incidentes, mantendo o foco na resolução.

Um desenvolvedor em dois contextos
Não há conflito em um profissional participar simultaneamente de um time Scrum (para desenvolvimento) e outro Kanban (para sustentação), obviamente que com queda da produtividade.
A chave está na clareza de prioridades:

No **Scrum**, o foco é o Sprint Goal e a entrega incremental;
No **Kanban**, a urgência determina a ação, com flexibilidade para interrupções.

Em empresas de software onde as pessoas são pegas a laço no corredor , são adicionadas de última hora em reuniões, as cerimônias prescritivas (como Sprint Planning e Daily Scrum) impõem disciplina , ajudando equipes a manter ritmo e foco.

A estrutura do Scrum, com sprints curtos e metas claras, força a decomposição de produtos em etapas gerenciáveis, reduzindo a ansiedade do "tudo para ontem" e promovendo entregas consistentes.

3-Tudo depende do tamanho do seu sapato.

Ao mencionar sustentação, estou me referindo a empresas que realizam dezenas, centenas ou até milhares de atendimentos por dia, aquelas que atendem usuários finais com conhecimentos limitados, por exemplo.

Para sustentações de menor escala, cuja negociação pode ser feita diretamente com o Product Owner, o Scrum dá conta do recado.
Além disso, o Scrum não impede a realização de horas extras; o que realmente importa é usar o bom senso.

Conclusão: Escolha Contextual, Scrum e Kanban são ferramentas complementares, não excludentes. Enquanto o Scrum revolucionou o desenvolvimento de software com sua estruturação e ciclos iterativos, o Kanban oferece agilidade para operações contínuas. A evolução do Scrum, incorporando práticas Lean, prova que a adaptação é essencial. Para organizações brasileiras, onde acultura ainda é um desafio, o Scrum traz ordem e o compasso.

Já o Kanban, com sua simplicidade, resolve o caos da sustentação. Ambos coexistem sob o guarda-chuva ágil, garantindo que inovação e operação avancem em sincronia.

Comparativo de frameworks de escalonamento do Scrum .

A adoção de metodologias e frameworks ágeis tem se tornado cada vez mais comum nas empresas de tecnologia, impulsionando a entrega de valor e a satisfação do cliente. No entanto, à medida que os times crescem e os projetos se tornam mais complexos, surge a necessidade de frameworks que suportem a escalabilidade. Neste contexto, frameworks como Nexus, Large Scale Scrum (LeSS) e Scaled Agile Framework (SAFe) ganharam popularidade. Aqui, vamos explorar as características de cada um e discutir por que a empresa SQLive optou pelo Nexus.

Nexus

O Nexus é um framework desenvolvido pela Scrum.org para conectar múltiplos times Scrum, permitindo que trabalhem em sincronia em um único Produto. Ele mantém a essência do Scrum, adicionando apenas os elementos necessários para a integração entre times.

Características do Nexus:
-Simples e Minimalista: O Nexus não cria papéis novos além dos definidos no Scrum, mantendo a estrutura de time Scrum tradicional.

-Foco no Produto: O Nexus é centrado no Produto e no valor entregue ao cliente, garantindo que todos os times trabalhem em direção ao mesmo objetivo.

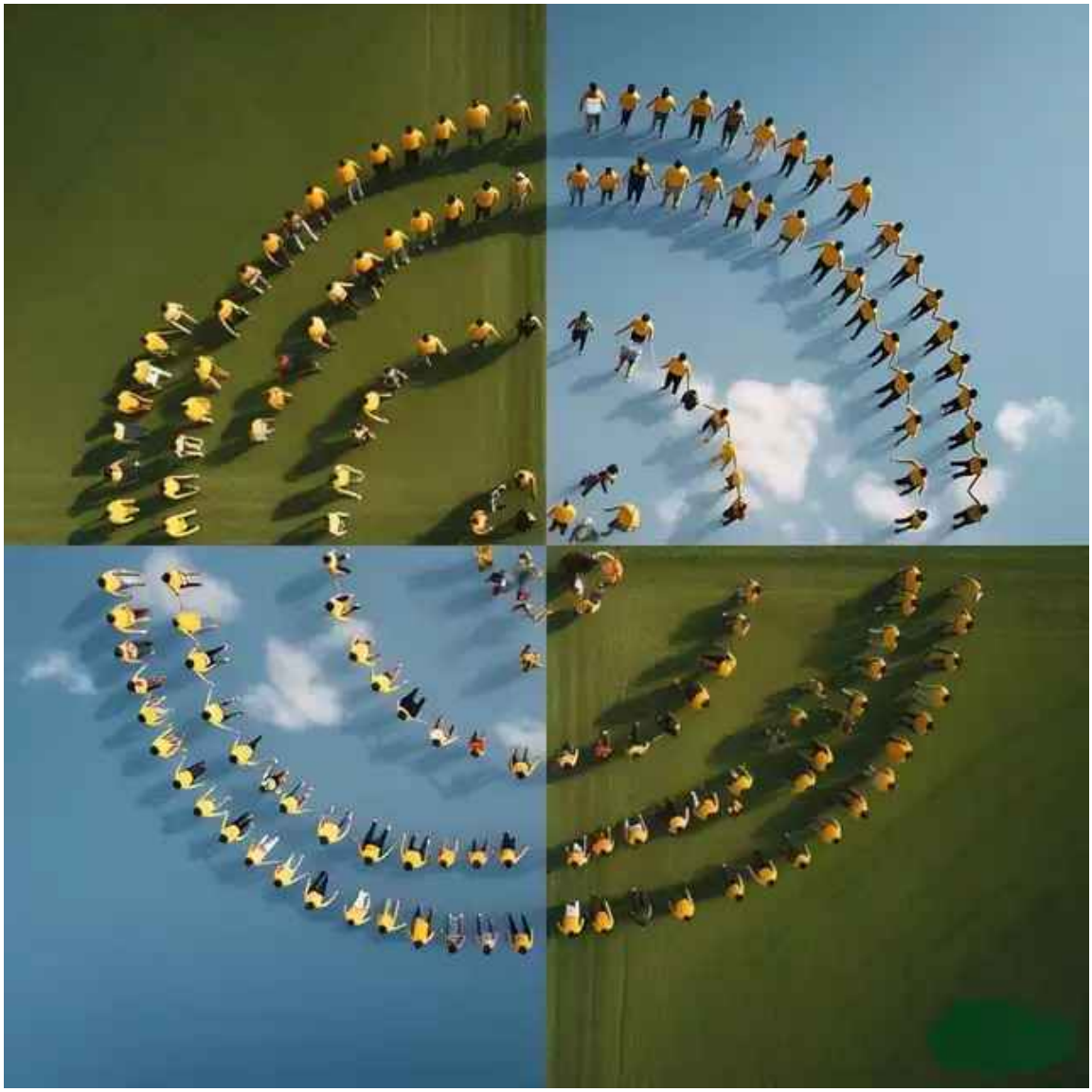
-Integração de Times: O Nexus define três artefatos adicionais: o Backlog do Produto Nexus, o Incremento Nexus e o Plano de Integração Nexus.

-Cerimônias: O Nexus adapta as cerimônias do Scrum para acomodar a escala, incluindo a Planejamento Nexus, Daily Scrum Nexus, Revisão Nexus e Retrospectiva Nexus.

Promessas do Nexus:
-Facilitar a transição para times já familiarizados com o Scrum.

-Garantir a implementação de práticas Scrum consistentes em todos os times.

-Fomentar a colaboração e a integração entre times.



Large Scale Scrum (LeSS)

O LeSS é um framework que aplica os princípios do Scrum em larga escala, com o mínimo de mudanças possíveis. Ele se baseia na ideia de que muitos dos problemas enfrentados pelos times ágeis em crescimento podem ser resolvidos aplicando mais rigorosamente os princípios do Scrum.

Características do LeSS:

-Foco na Estrutura Scrum: O LeSS mantém a estrutura Scrum tradicional, com times Scrum trabalhando em um único Backlog do Produto.

Princípios LeSS:
- O framework define princípios como "Feature Teams", "Component Teams são ruins" e "Um Backlog do Produto".

-Adaptação de Cerimônias: O LeSS adapta as cerimônias Scrum para acomodar a escala, incluindo a Planejamento LeSS e Revisão LeSS.

Promessas do LeSS:
-Promover a colaboração e a responsabilidade compartilhada entre times.
-Garantir que todos os times trabalhem em direção ao mesmo objetivo.
-Fomentar a melhoria contínua e a adaptação.

Scaled Agile Framework (SAFe)

O SAFe é um framework abrangente para a adoção de práticas ágeis em larga escala. Ele fornece uma estrutura detalhada para gerenciar múltiplos times ágeis, incluindo times de desenvolvimento, gerenciamento de produtos e outras funções.

Características do SAFe:
-Estrutura de Níveis: O SAFe define quatro níveis: Time, Programa, Grande Solução e Portfólio.

-Cerimônias e Práticas: O SAFe inclui cerimônias como PI Planning, Daily Scrum e System Demo.

-Foco na Arquitetura: O SAFe enfatiza a importância da arquitetura e da solução técnica.

Promessas do SAFe:

-Fornecer uma estrutura abrangente para gerenciar múltiplos times ágeis.

-Garantir o alinhamento entre times e objetivos estratégicos.

-Fomentar a colaboração e a entrega de valor.

Todos os três frameworks - Nexus, LeSS e SAFe -buscam escalar suas práticas ágeis. No entanto, eu prefiro o Nexus devido à sua simplicidade e aderência estrita aos princípios do Scrum, sem novos papéis,ou parafernalhas.

Conclusão

A escolha de um framework ágil escalável é estratégica e impacta diretamente a eficácia e o desempenho dos times. LeSS e SAFe têm seus méritos, mas a SQLive encontrou no Nexus a solução ideal: simplicidade e alinhamento total com o Scrum. Com essa decisão, a empresa escalou agilidade sem burocracia, mantendo a essência do Scrum sem introduzir funções ou etapas desnecessárias.

Sinceramente tenho minhas restrições contra frameworks que tentam tornar o Scrum "mais palatável para quem vem do waterfall', e para quem acredita em complexas estruturas de controle , e essa definitivamente não é a abordagem da Scrum.org.

A fantástica fábrica de certificados.

O que eu penso desse monte de frameworks e certificações fora da Scrum.org e da Scrum Alliance?

Existem duas certezas na vida: a morte, o lançamento de novos frameworks JavaScript e certificações Scrum.

A maioria se propõe a consertar o que não está quebrado, tornar o Scrum mais palatável para quem não acredita em agilidade e ganhar muito dinheiro.

Eu pessoalmente apoio a Scrum.org, pois a considero mais rigorosa e justa.

A Scrum.org cobra por certificações, exigindo 85% de acerto, e não por cursos nos quais o certificado sai praticamente em uma máquina de chiclete.

Recertificar alguém depois de dois anos, quando não houve alterações no guia do Scrum, chega a ser ofensivo.

Por acaso, quando lançaram o aquecedor de vidros, chamaram todos os motoristas para tirar a carteira de motorista novamente?

Conclusão

Não gaste seu dinheiro com certificações de qualidade duvidosa, elas não trarão resultado prático para sua carreira.

O Time Scrum e a IA: Oportunidades e desafios.

A Inteligência Artificial (IA) está revolucionando o modo de operação das equipes de desenvolvimento de software, e o framework Scrum também é afetado por essa mudança. Através da automação de tarefas rotineiras, da análise preditiva e da otimização nas decisões, a IA tem potencial para aumentar a produtividade das equipes, ao mesmo tempo que apresenta novos desafios para os times ágeis. Abordaremos como a IA influencia cada função dentro do Scrum (Scrum Master, Product Owner e Desenvolvedores), os desafios , e de que maneira essa tecnologia pode ser aplicada para melhorar a eficiência.

1. Impacto da IA nos Papéis do Scrum

a) Scrum Master

O Scrum Master tem a missão de eliminar obstáculos e assegurar que a equipe siga os princípios ágeis. A inteligência artificial pode ajudar nesse trabalho de diferentes formas.

-Automação de Cerimônias: Ferramentas de IA podem agendar reuniões, gerar resumos automáticos de Daily Standups e identificar padrões de comunicação que podem ser melhorados.

-Análise de Retrospectivas: Plataformas como o **Fireflies.ai** ou **Otter.ai** podem transcrever e analisar discussões, identificando temas recorrentes e sugerindo melhorias.

-Detecção de Riscos: Algoritmos podem prever possíveis atrasos baseados em dados e métricas como WIP,Troughput, Cycle time e capacity.]

-IA´S podem criar roteiros de dinâmicas: Baseando-se em dinâmicas constantes nos livros e artigos), elas não tem a criatividade para criar dinâmicas novas.

Desafio: O Scrum Master deve aprender a entender os dados produzidos pela inteligência artificial e assegurar que as recomendações estejam de acordo com os princípios ágeis, evitando ficar muito dependente da automação.

Tente um dia gerar um artigo por uma IA e você verá a quantidade de dados errados e visões ultrapassadas:Opiniões como:Só se pode entregar na review, usar velocity ao invés de troughput para cálculo de Wip, e outras bobagens.

b) Product Owner (PO)

O PO é responsável pelo backlog e pela priorização de histórias. A IA pode trazer os seguintes benefícios :

-Priorização Inteligente: Ferramentas como **Aha!** ou **Jira AI** podem sugerir a ordem do backlog com base em dados de mercado, feedback de usuários e custo de desenvolvimento.

-Geração de User Stories: Os diversos modelos de linguagem podem ajudar a escrever descrições mais claras.

-Análise de Feedback: IA pode processar grandes volumes de feedback de clientes (reviews, suporte) para priorizar melhorias.

Desafio: O PO precisará validar as sugestões da IA, mantendo o foco no valor real para o usuário e não apenas em métricas automatizadas(IA analisa dados, e tendências, ela não tem feeling, ou criatividade).

c) Time de Desenvolvimento

Para desenvolvedores, QA´S (um desenvolvedor especializado em testes), designers(um desenvolvedor especializado em UI), a IA pode acelerar tarefas repetitivas e melhorar a qualidade.

-Geração de Código: Ferramentas como **GitHub Copilot** e **Amazon CodeWhisperer** sugerem trechos de código, reduzindo tempo de desenvolvimento.

-Testes Automatizados: IA pode criar e executar testes baseados em mudanças no código, identificando bugs mais rapidamente.

-Otimização de DevOps: Sistemas como **Datadog** e **New Relic** usam IA para detectar anomalias em tempo real.

Desafio: A equipe deverá examinar minuciosamente o código produzido por inteligência artificial para prevenir questões de segurança e qualidade. Ademais, a dependência de ferramentas pode diminuir a compreensão mais abrangente do sistema.

Tenho visto muito código SQL medíocre gerado pelas IA´s, por exemplo.

2. Novos Desafios Introduzidos pela IA no Scrum

-Ética e Viés: Decisões baseadas em IA podem refletir vieses.

-Perda de Toque Humano: Excessiva automação pode reduzir a colaboração e a criatividade do time.

-Manter o Foco Ágil: Lembrar que a IA é uma ferramenta de produtividade, nada mais que isto, quem será cobrado por eventuais erros, é você.

MINHA OPINIÃO(2025)

Na minha visão, atualmente, há uma supervalorização das tecnologias de inteligência artificial existentes atualmente. Infelizmente, o setor de IA e certos consultores sem ética estão propagando a ideia de que, em um futuro próximo, não haverá mais necessidade alguma de desenvolvedores.
Prevejo que a realidade será dura, resultando em muitos erros nas implementações, código vulnerável, interfaces rígidas, além de perdas financeiras e desilusões.

A inteligência artificial estará em constante avanço e de forma acelerada. Atualmente, sua presença é essencial, embora grande parte de suas funções consista em acessar informações disponíveis para os humanos

(como no Stack Overflow, por exemplo) e adaptar códigos de acordo com as solicitações recebidas.

Ela faz, o que os programadores têm feito ao longo de várias décadas, mas ainda tem muitas dificuldades com o ajuste fino.

Tenho visto Ias sugerindo soluções medíocres na minha área de expertise(SQL).

Não existe APP mais ou menos, um APP feio e pouco flexível não será usado, isso vale para cada item da arquitetura de um sistema.
Não existe sistema um pouco seguro também.

E hoje as Ias estão longe da qualidade de programadores bons.

Segundo Mark Zuckerberg:
"Em 18 meses a codificação será feita por IA, será melhor que o trabalho da maioria dos engenheiros. ".

Para mim é falácia, tanto no prazo, quanto do tipo de codificação que as Ias serão capazes de criar a médio prazo, estamos em junho de 2025, me cobrem em dezembro de 2026,e em Junho de 2036.

A qualidade é fundamental no Scrum e pode ser aprimorada ao longo do tempo, mas deve sempre se basear em fundamentos sólidos, nunca partir de uma base medíocre, ou que não seja 100% confiável.
Resumindo a IA é uma ferramenta útil, mas ainda está longe de substituir 100% o desenvolvedor.

Não faça parte da maioria citada pelo dono da Meta, seja uma referência.
Não lute contra a IA, seja o melhor na sua utilização.

Conclusão

A Inteligência Artificial possui a capacidade de transformar a maneira como os times Scrum operam, proporcionando aumento de produtividade, mas também apresentando novos desafios. O segredo reside em balancear automação e pensamento crítico, assegurando que a tecnologia atue como um instrumento facilitador, e não como uma substituta para a criatividade e a cooperação humanas. Equipes que souberem incorporar a Inteligência Artificial de maneira estratégica estarão em melhor posição para oferecer produtos mais inovadores e de maior valor agregado.

A Evolução dos frameworks ágeis e a necessidade de adaptação na era dos microsserviços

Frameworks ágeis e a arquitetura dos anos 90

A maior parte dos frameworks ágeis, incluindo o Scrum, emergiu na década de 1990, período em que o desenvolvimento de software ainda era amplamente moldado por arquiteturas monolíticas. Nesse cenário, um incremento do produto normalmente era associado a uma única implementação de novos itens em um sistema que continha uma gama enorme de funcionalidades. A entrega seguia um fluxo mais linear, e a Sprint Review servia como um ponto de verificação focal para validação antes da liberação em produção.

Microsserviços e a fragmentação do incremento

Com o advento dos microsserviços, a dinâmica mudou. Uma única equipe Scrum frequentemente trabalha em múltiplos microsserviços simultaneamente, o que significa que, em uma mesma sprint, vários incrementos independentes podem estar prontos para produção em momentos diferentes. Nesse cenário, esperar a *Review* para liberar um microsserviço já testado e validado pode não fazer sentido – especialmente se ele não tem dependências críticas com outros itens em desenvolvimento.

A importância do gerenciamento de dependências no Sprint Backlog

Quando times trabalham com microsserviços, o *Sprint Backlog* passa a ter um nível maior de complexidade devido às interdependências entre tarefas. Se um microsserviço depende de uma API que está sendo desenvolvida em paralelo, liberá-lo antes da conclusão dessa API pode gerar inconsistências. Por isso, é essencial mapear e controlar essas relações para evitar:

-Deploys parciais que quebram funcionalidades

-Retrabalho devido a integrações mal planejadas

-Atrasos causados por gargalos em dependências críticas

Visualizando Dependências: Gráficos e Ferramentas

Para auxiliar nesse gerenciamento, alguns mapas é gráficos são de extrema utilidade:

Gráfico de Dependências (Dependency Graph)

-Mostra as relações entre os itens do *Sprint Backlog* em formato de nós e arestas.

-Identifica microsserviços que podem ser liberados antecipadamente (sem bloqueios) e aqueles que precisam aguardar outros componentes.

Matriz de Dependências

-Uma tabela que cruza serviços e indica quais têm relações de dependência (ex.: "Serviço A precisa do endpoint do Serviço B").

Roadmap de Deploys

-Um cronograma visual que define a ordem ideal de liberação com base nas interdependências.

Conclusão

Os frameworks ágeis tradicionais precisam ser adaptados para cenários de microsserviços, onde entregas incrementais independentes são comuns. A Sprint Review mantém seu valor como cerimônia de revisão do que foi feito, e de realinhamento, mas times devem ter autonomia para liberar em produção quando fizer sentido – desde que as dependências estejam claramente mapeadas. Ferramentas visuais, como gráficos e matrizes, são fundamentais para evitar riscos e garantir que a agilidade não se perca em meio à complexidade arquitetural moderna.

Não, não era tudo mato!

Uma mentira que vem sendo propagada, de boca em boca, nos últimos anos, é que antes do manifesto ágil só se usava Waterfall.

Em minha vida profissional, nunca precisei fazer um fluxograma.

Realmente, quando trabalhei para bancos no início da década de 90 eu usei waterfall, e o pior é que funcionava, já que os bancos não eram startups que estavam inventando algo revolucionário, eles sabiam o que queriam , na maioria das vezes só desenvolvíamos o mesmo software que a concorrência tinha, não aviam surpresas.
As metodologias ágeis ajudam a domar a incerteza, em equipes que exploravam novos mercados.

Quando montei minha Software House, no final de 1993, comecei a explorar nichos do mercado como empresas de segurança e correio privados, planos de saúde ,agências de publicidade, construtoras do ramo de fundações, e fábricas de formulário contínuo, empesas que na época não tinham sistema nenhum.

Eu tinha que lidar com clientes(nem conhecia o termo stakeholder), eu bancava o desenvolvimento e era dono dos direitos autorais, ou seja, eu não sofria nenhum tipo de pressão por prazos, a concorrência era pequena, quando eu achava que o produto estava pronto para o mercado usava os vendedores da fábrica de formulários contínuos para vendê-los.

Foi muito fácil começar, até as encomendas de software chegarem, e eu ter que começar a contratar programadores, como diria o Zagalo:”**Aí fomos surpreendidos novamente!”**

Era muito difícil escalonar a minha metodologia “DOITNOWBRO”, não que eu não tentasse.
Meu programadores tinham que usar meu código,minha padronização, eu tentava supervisionar a qualidade, passar diretrizes, inspecionar o que era feito, mas nem sempre funcionava.

Alguns eram bons “mini Product owners/desenvolvedores”, outros não. Porque o desenvolvedor as vezes era bonzinho demais e aceitava cada miníma ideia sugerida, mesmo que não acrescentasse valor nenhum ao produto, outros entendiam errado o que o cliente queria, outros não entendiam que “softwares empacotados” tinham que funcionar para várias empresas.

Desnecessário dizer que minha vida social acabou no período, trabalhava no mínimo 15 horas por dia.

Tentei inventar a “Daily” ,tentava pegar o pessoal a laço ,perguntava se alguém tinha alguma dificuldade, passava experiências “do front”, talvez isso tenha sido o segredo do sucesso relativo que tive na época.

Mas era quase impossível juntar os programadores, alguns estavam ao telefone, outros estavam correndo para finalizar serviços.

Metodologias como o Scrum não vieram só acabar com o Waterfall, vieram trazer paz e disciplina para quem era averso a ele também.

Os pilares e valores do Scrum ajudaram os desenvolvedores a entender o valor da qualidade e da melhoria contínua.

A Daily, um alinhamento curto, sempre no mesmo horário, com a mesma duração, no mesmo lugar.
Ninguém assume outro compromisso para o horário, e já vem sabendo o que têm que dizer de maneira resumida.

As histórias de usuário substituíram as antigas especificações funcionais do pessoal do Waterfall, a assertiviade aumentou muito através do Product Backlog ,com sua transparência e disponibilidade.

A Review faz com que as coisas não saiam do controle, melhorou a vida de quem usava waterfall, e também de quem utilizava as "metodologias" ágeis da época.

Quem critica o TimeBox da Daily não entendeu o sentido dela.

Os TimeBoxes da Planning, Review e Retrospectiva são seus tempos máximos de duração, muitas vezes não levam nem 30% do tempo.

Ou seja, o tempo gasto com a "parte burocrática" é infinitamente menor do que o gasto com a tralha inútil do Waterfall.

Em Software houses maiores sempre existiu uma figura que não programava, o gerente de desenvolvimento, só que ele se preocupava com cobrança e não com dissiminação de valores que levam a qualidade e melhorias de processos.

Recentemente, uma pessoa relacionada à Caixa Econômica Federal disse que os Scrum Masters são inúteis, quase inúteis eram os antigos preenchedores de relatórios, planilhas e cronogramas da ilha da fantasia.

Ser Scrum Master em uma empresa estatal, no Brasil, não deve ser fácil.

Lembrando que: em empresas pequenas pode haver acúmulo de papéis, um programador pode fazer o papel de Scrum Master também.

Vale lembrar que antes do Scrum existiram várias tentativas como XP e RUP, entre outras, mas nunca me simpatizei com nenhuma delas.

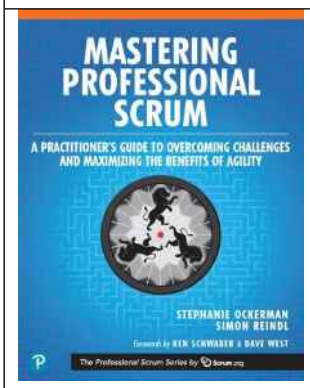
Conclusão

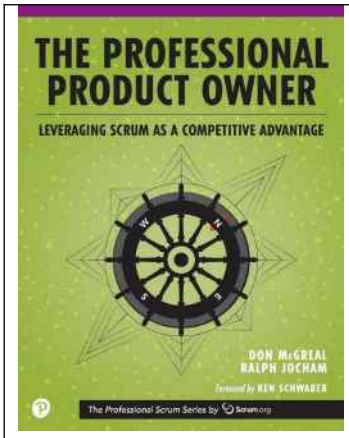
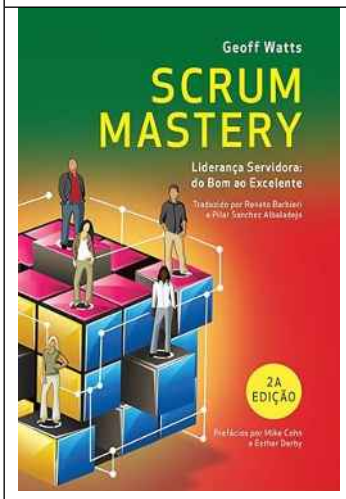
A agilidade sempre existiu no Brasil, o que quase não existia era metodologia.

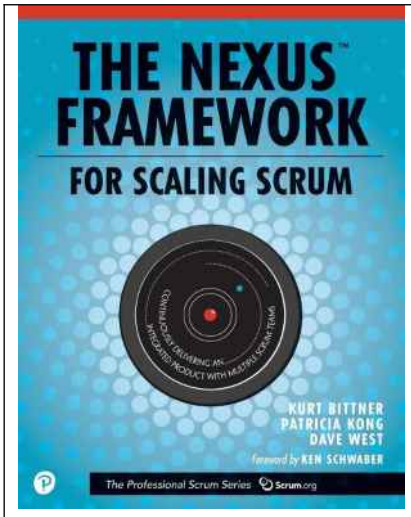
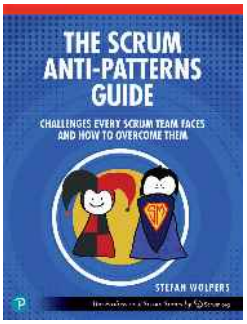
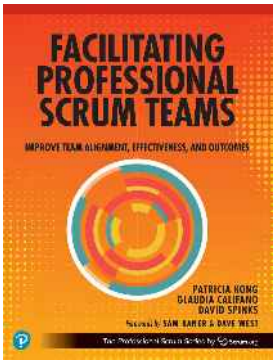
Aqui estão alguns livros que eu recomendo sobre Scrum.

A maioria das obras disponíveis atualmente não está atualizada em relação ao atual Guia do Scrum (2020). A seguir, apresento minha perspectiva sobre livros que acredito acrescentariam ao conhecimento de quem deseja atuar com Scrum, em seus diversos papéis.

Todos são em inglês, se você quer trabalhar com Scrum, vai precisar dominar o idioma, ou se virar com algum tradutor.

	Applying Scrum with Kanban: A Pointless Book! -2021-Andy Hiles Livro que aboarda as técnicas de Scrum com Kanban, mostra como os storypoint são ineficientes e mostra de maneira divertida como montar e visualizar o workflow.Leading and lagging indicators,definition of workflow,etc.
	**Mastering Professional Scrum: A Practitioner's Guide to Overcoming Challenges and Maximizing the Benefits of Agility (The Professional Scrum Series) (English Edition)** Dicas do dia a dia,sobre remoção de impedimentos,exemplos claros sobre definition of done,Como montar um grande time Scrum.

	**The Professional Product Owner: Leveraging Scrum as a Competitive Advantage Capa comum – Ilustrado, 4 junho 2018** Edição Inglês por Don McGreal (Autor), Fala das responsabilidades do product owner dentro do framework Scrum(como deve agir em cada cerimônia,sobre os tipos de PO e como se memolhar enquanto profissional,hard skills e soft skills.Mostra que o P.o é um maximizador de valor e não um gerente burocrático de projetos.
	**SCRUM MASTERY Liderança Servidora: do Bom ao Excelente-Geoff -Watts-2a edição-Português** Estratégias para tornar as reuniões Scrum mais produtivas As qualidades e competências de um ótimo Scrum Master Técnicas para melhorar o desempenho da equipe Dicas para fortalecer o envolvimento da equipe Formas de incentivar o time como incentivar a colaboração e a criatividade

	**The Nexus Framework for Scaling Scrum: Continuously Delivering an Integrated Product with Multiple Scrum Teams Capa comum – Ilustrado, 17 dezembro 2017** -Descreve com detalhes a implementação do Nexus Framework-Um framework para escalonamento do Scrum.Como gerenciar as dependências de vários times scrum trabalhando sob o comando de um só P.O e um só backlog,desde a definition of done compartilhada até o funcionamente da Equipe de Integração do Nexus (NIT).
	The Scrum Anti-Patterns Guide O livro aborda os antipadrões mais comuns que aparecem em projetos que usam Scrum. Entre eles, estão problemas na comunicação, dificuldades na elaboração da (Definition of Done), ausência de transparência e do respeito pelos princípios do Scrum.
	Facilitating Professional Scrum Teams Facilitando o alinhamento, facilitando o Planejamento da Sprint,Facilitando a Daily Scrum,facilitando a dinâmica da equipe. Técnicas para conduzir reuniões diárias produtivas, seja para atualizações de status ou para resolver problemas rapidamente. Estratégias para lidar com desafios como conflitos e melhorar a tomada de decisão

Se esse Livreto lhe foi útil considere doar qualquer quantia(R$ 30?.) para alguma das associações que ajudam as crianças autistas no Brasil, O autismo hoje cresceu 3000% em relação a décadas atrás, as causas continuam em estudo, mas as necessidades das crianças não podem esperar. Crianças autistas carecem não só de auxílio psiquiátrico, mas também, nutrológico e educacional.

Para doar via PIX para uma associação de crianças autistas, você pode procurar a chave PIX da instituição que deseja apoiar, como a AUMA (Associação dos Amigos da Criança Autista) ou a ABADS (Associação Brasileira de Assistência e Desenvolvimento Social). A AUMA utiliza a chave PIX 38.893.038/0001-03 e a ABADS a chave doe@abads.org.br.

Sobre o Autor:

Marcello Eduardo de Oliveira Dias , começou seu estudos em T.I EM 1986 nos laboratórios do Colégio Comercial Brasil da Vila Carrão-SP,É Pós-Graduado em Engenharia de Software pela Bráz Cubas.
Possui as certificações:PSM1,PSMPO1,PSK1 e SPS da Scrum.org

Durante sua carreira passou por empresas como Mais Soft, Aon, Fidelity, Brasfond, SQLive e Ultralub.

https://www.linkedin.com/in/marcello-eduardo-de-oliveira-dias-721201362/

INCREMENTO
INCREMENTO
INCREMENTO
IA
NEXUS
LEAN
SCRUM
KANBAN
TRANSPARÊNCIA
INSPEÇÃO
ADAPTAÇÃO

SCRUM
MASTER
BOX